NASA / TM-1999-209108

Design of the Wind Tunnel Model Communication Controller Board

William C. Wilson
Langley Research Center, Hampton, Virginia

March 1999

The NASA STI Program Office ... in Profile

Since its founding, NASA has been dedicated to the advancement of aeronautics and space science. The NASA Scientific and Technical Information (STI) Program Office plays a key part in helping NASA maintain this important role.

The NASA STI Program Office is operated by Langley Research Center, the lead center for NASA's scientific and technical information. The NASA STI Program Office provides access to the NASA STI Database, the largest collection of aeronautical and space science STI in the world. The Program Office is also NASA's institutional mechanism for disseminating the results of its research and development activities. These results are published by NASA in the NASA STI Report Series, which includes the following report types:

- TECHNICAL PUBLICATION. Reports of completed research or a major significant phase of research that present the results of NASA programs and include extensive data or theoretical analysis. Includes compilations of significant scientific and technical data and information deemed to be of continuing reference value. NASA counterpart of peer-reviewed formal professional papers, but having less stringent limitations on manuscript length and extent of graphic presentations.

- TECHNICAL MEMORANDUM. Scientific and technical findings that are preliminary or of specialized interest, e.g., quick release reports, working papers, and bibliographies that contain minimal annotation. Does not contain extensive analysis.

- CONTRACTOR REPORT. Scientific and technical findings by NASA-sponsored contractors and grantees.

- CONFERENCE PUBLICATION. Collected papers from scientific and technical conferences, symposia, seminars, or other meetings sponsored or co-sponsored by NASA.

- SPECIAL PUBLICATION. Scientific, technical, or historical information from NASA programs, projects, and missions, often concerned with subjects having substantial public interest.

- TECHNICAL TRANSLATION. English-language translations of foreign scientific and technical material pertinent to NASA's mission.

Specialized services that complement the STI Program Office's diverse offerings include creating custom thesauri, building customized databases, organizing and publishing research results ... even providing videos.

For more information about the NASA STI Program Office, see the following:

- Access the NASA STI Program Home Page at *http://www.sti.nasa.gov*

- E-mail your question via the Internet to help@sti.nasa.gov

- Fax your question to the NASA STI Help Desk at (301) 621-0134

- Phone the NASA STI Help Desk at (301) 621-0390

- Write to:
  NASA STI Help Desk
  NASA Center for AeroSpace Information
  7121 Standard Drive
  Hanover, MD 21076-1320

NASA / TM-1999-209108

Design of the Wind Tunnel Model Communication Controller Board

William C. Wilson
Langley Research Center, Hampton, Virginia

National Aeronautics and
Space Administration

Langley Research Center
Hampton, Virginia 23681-2199

March 1999

Available from:

NASA Center for AeroSpace Information (CASI)
7121 Standard Drive
Hanover, MD 21076-1320
(301) 621-0390

National Technical Information Service (NTIS)
5285 Port Royal Road
Springfield, VA 22161-2171
(703) 605-6000

Design of the Wind Tunnel Model Communication Controller Board.

William C. Wilson
NASA Langley Research Center
M/S 488
Hampton, VA 23681-2199
w.c.wilson@larc.nasa.gov

Abstract

The NASA Langley Research Center's Wind Tunnel Reinvestment project plans to shrink the existing data acquisition electronics to fit inside a wind tunnel model. Space limitations within a model necessitate a distributed system of Application Specific Integrated Circuits (ASICs) rather than a centralized system based on PC boards. This thesis will focus on the design of the prototype of the communication Controller board. A portion of the communication Controller board is to be used as the basis of an ASIC design. The communication Controller board will communicate between the internal model modules and the external data acquisition computer. This board is based around an FPGA, to allow for reconfigurability. In addition to the FPGA, this board contains buffer RAM, configuration memory (EEPROM), drivers for the communications ports, and passive components.

Acknowledgments

I would like to acknowledge the following people for their efforts on this project. Dr. Robert Hodson for his guidance and patience, Duane Armstrong for his efforts on system design, Darren Boyd for his work on the Internal Simulator board, and Denise Linthicum for her work on Host Interface Board.

Table of Contents

List of Figures

List of Tables

Nomenclature

ACK	Acknowledge Command
ADC	Analog to Digital Converter
AMD	Advanced Model Development
AoA	Angle of Attack
BDDU/CPA	Balance Dynamic Data Unit/Critical Point Analyzer
Bps	Bits per second
ASIC	Application Specific Integrated Circuit
CLB	Configurable Logic Block
CMOS	Complimentary Metal Oxide Silicon
DAC	Digital to Analog Converter
DAS	Digital Acquisition System
DUT	Device Under Test
EOF	End Of Frame
EPROM	Erasable Programmable Read Only Memory
EEPROM	Electrically Erasable Programmable Read Only Memory
FIFO	First In First Out
FPGA	Field Programmable Gate Array
HDL	Hardware Description Language
ICD	Interface Configuration Document
MCM	Multichip Module

MEM	Micro- Electro-mechanical Machine
NASA	National Aeronautics and Space Administration
NRZ	Non Return to Zero
PC	Printed Circuit
PGA	Pin Grid Array
Op Amp	Operational Amplifier
RAM	Random Access Memory
ROM	Read Only Memory
SIO	Serial Input Output
SRAM	Static Random Access Memory
V	Volts
VHDL	VHSIC Hardware Description Language
VHSIC	Very High Speed Integrated Circuits
ZIF	Zero Insertion Force

Chapter 1 Introduction

The current wind tunnel approach to data acquisition, makes use of several interface boxes to convert analog signals that come from the model to digital signals for use by a Modcomp computer. These analog signals must run several hundred feet from the model before they are converted to digital. The long runs of the signal wires are bundled together to create cables. It takes a considerable amount of time to pull new cables for each model. If a standardized cable system was used it would eliminate the need to pull new cables for each model. Also, if the system digitized the signals at the source this would eliminate some of the interface boxes between the model and the Modcomp computer.

NASA Langley Research Center is involved in a re-investment plan. This plan allows money to be invested in facility engineering tasks that will give high returns (in terms of savings) in the future. One such project is the Advanced Model Development (AMD) Project. This project plans to improve wind tunnel test effectiveness by reducing the mechanical loading on the model, reducing the model setup time, and improving the accuracy of data available to the researchers.

1.1 Goals of the Advanced Model Project

The main goal of the AMD is to shrink the existing data acquisition electronics to fit inside the wind tunnel model. Towards this end, it will be necessary to move from printed circuit wiring boards to Application Specific Integrated Circuits (ASICs). Space limitations within a model necessitate a distributed system rather than a centralized one. In this type of environment, the sensors will be clustered into groups of up to eight sensors for a single DAS (Digital Acquisition System) unit. Since each of the sensors would connect directly to a DAS ASIC, the board would be called a module. The separate modules would be tied individually to a communication controller ASIC. By using a serial bus, the system would attain a high degree of flexibility. The number of sensors could then be changed efficiently by the addition/removal of sensor modules. A minimal amount of wiring changes (power connections, communication bus) would be necessary.

Communication for the model data acquisition system will be managed by a communication Controller ASIC to facilitate data gathering and communications tasks. The sensor modules will be configured from a host computer by way of the communication Controller ASIC. After data has been collected and buffered, the communication Controller will send the data to the host computer using a fiber optic cable. It will be necessary for the host computer to have a fiber optic interface.

The main types of sensors found in a wind tunnel model fall into one of five types, temperature, pressure, angle of attack (AoA), shear stress sensors, and balance (force) measurements. Data will come as raw analog signals directly from the sensors. The pressure, AoA, shear stress, and balance data acquisition sensors will need signal conditioning. This means input filtering, amplification, multiplexing, and measurement by a whetstone bridge or other mechanism (constant current source). Once input filtering is complete, the modules will convert the analog signal to a digital format and send the data to the communication Controller ASIC. Each type of sensor module will be implemented in its own ASIC.

1.2 Thesis Topic

This thesis will focus on the design of the wind tunnel model Communication Controller prototype board for an ASIC. A portion of the communication Controller board is to be used as the design of an ASIC. The communication Controller board communicates between the sensor clusters, internal to the model and the external data acquisition computer. This board is based around a Xilinx FPGA to allow for reconfigurability. In addition to the FPGA chip this board contains buffer RAM, configuration information in a non-volatile memory (EEPROM), drivers for the internal and external communications ports, and passive components. This module uses the internal communication bus to talk to the other sensor modules.

The most challenging portion of this endeavor is to create a design that fits the within the space limitations of the model, while achieving the functionality needed to accomplish the data acquisition and communication tasks.

Chapter 2 System Architecture

There are several reasons why the design is broken up into separate modules. One, dividing the system allows for additions/removal of sensor modules with minimal impact. Two, any future sensors/modules can be added by interfacing to the internal bus instead of re-designing the entire data acquisition system. Three, small spaces can be carved out in areas distributed throughout the inside of the model for the electronics, instead of trying to find enough space for the combined system. Four, a distributed system provides a path for alternate methods of communication, such as a fiber optic cable or an interface for a radio frequency telemetry system. Finally five, the system can be distributed and still allow for reconfigurability and remote configuration. See figure 1 for a block diagram of the data acquisition system.

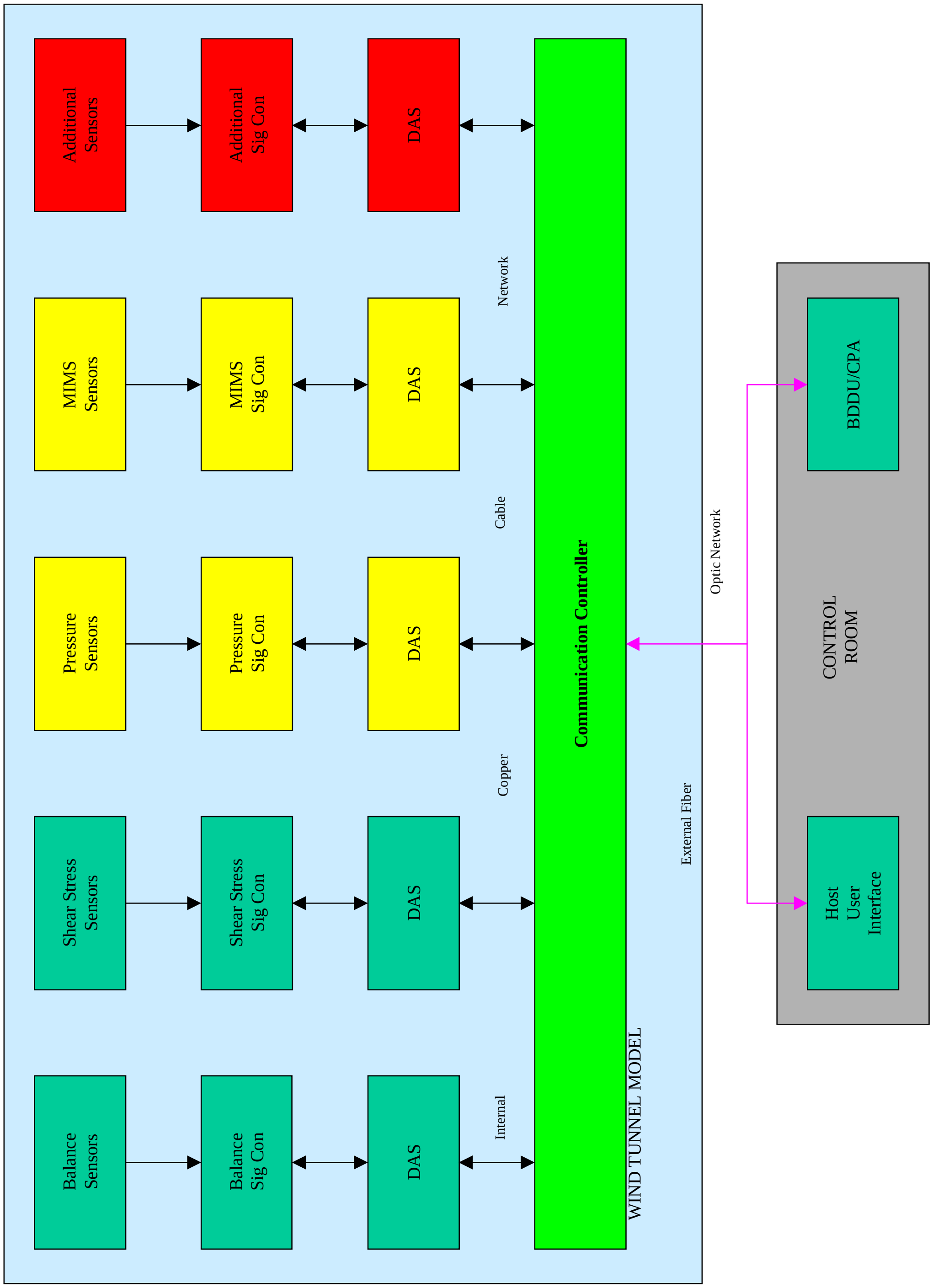


Figure 1 System Block Diagram

The general system architecture is a distributed one. Each type of sensor will have a signal-conditioning module; in addition, there will be a data acquisition module for every eight sensors that make up a cluster, and a single communication Controller module. Each of the modules for the prototype has been implemented on a printed circuit wiring board.

The host interface is designed to perform three main tasks. First, it converts the fiber optic signals from the model to electrical signals for the control room's host computer system. Second, the host interface provides for acquisition and configuration of engineering data. Third, the system provides a second data path directly to the Balance Dynamic Data Unit/Critical Point Analyzer (BDDU/CPA) for safety features. In addition to the primary data collected during a wind tunnel run, there is engineering data about the condition of the model, sensors, or the model environment, that is monitored only and not stored. The host interface must convert the fiber optic signals to a form that can be used by the display terminal in the wind tunnel control room. The host interface must also provide a path for configuration data from the display terminal back to modules inside the model.

All of the modules residing within the model will find it necessary to use ASIC technology in the design for size reduction alone. As stated earlier the modules are based on a distributed scheme that will allow for the most amount of reconfigurability and versatility. The essential module to the system's versatility is the communication Controller module. The model data acquisition system is managed by a communication Controller module to facilitate data gathering and communications tasks. The external world communicates to the wind tunnel model using a single fiber optic cable. Command, configuration, and data will all use the same bus. The communication Controller module will in turn communicate to each of the sensor modules over a differential serial bus.

The sensor modules are configured from a host computer by way of the communication Controller module. After data has been collected and buffered from the sensor modules, the communication Controller sends the data to the external host interface.

The BDDU/CPA is a real-time data analyzer for safety concerns. The BDDU/CPA analyzes the six balance strain gages, looking for the conditions when the loads exceed the maximum limits. To do this job effectively it is necessary to use matrix manipulation to take all of the possible cross terms into account. If the loads do exceed the project maximums, then the model will be dropped to a zero angle of attack and the tunnel may be shut down. This is a separate safety function to be performed in parallel with data acquisition tasks. The fiber optic cable will have a 50/50 splitter in the transmission line from the model. Therefore, both the host interface and the BDDU/CPA will receive the same signals from the communication Controller module within the model.

The types of sensors found in a wind tunnel model fall mainly into one of five types, temperature, pressure, angle of attack (AoA), shear stress using hot film techniques, and balance (force) measurements. Sensor data comes in as raw analog signals directly from the sensors to the DAS module where it is conditioned, multiplexed, and converted to digital. Each DAS module has a means of communication with the communication Controller module.

The design of the DAS modules consists of two portions; one is a generic data acquisition portion made up of a communication interface and analog to digital conversion electronics. The second portion contains the signal conditioning needed for each type of sensor. This module is called the DAS (Digital Acquisition System) ASIC. As stated above, the DAS is used in conjunction with signal conditioning for the Shear Stress ASIC or Balance sensors. The DAS has a microcontroller core for communication and control of the ASIC. Eight channels of analog inputs are multiplexed into one analog to digital converter. The DAS module will provide 16-bit resolution, oversampling capability, and optionally some digital signal processing.

The six strain gauge bridges on the Balance would all be multiplexed together and sent to the DAS ASIC. All of the engineering data from thermocouples are multiplexed together and sent to the two remaining channels of the DAS ASIC. All of these functions would occur on the DAS ASIC.

The shear stress sensor is a hot film type of sensor made using MEMs (Micro-Electronic Machine) technology. The Shear Stress sensors also require signal conditioning. This means input filtering, amplification, multiplexing. All of these functions would occur on the DAS ASIC. Because of the high data rate, only two Shear Stress sensors are placed on a cluster. If the two sensors are orthogonal to each other, flow direction can be determined. Both channels are for the hot film sensors, which are sent to the DAS ASIC.

The AoA sensors need signal conditioning. This means input filtering, amplification, multiplexing, and measurement by a whetstone bridge or other mechanism (constant current source). All of these functions would occur on the DAS ASIC. On the DAS ASIC two channels of AoA data would exist, for the two accelerometers.

The pressure sensor module requires signal conditioning. This means input filtering, amplification, multiplexing, and measurement by a whetstone bridge or other mechanism (constant current source). All of these functions would occur on the DAS ASIC. The preferred type of pressure sensor for the sensor cluster approach is a MEMs (Micro-ElectroMechanical) pressure sensor.

In addition to the standard types of sensors mentioned, the data acquisition system must prove flexible enough to handle the additions of sensors proposed for the near future. New module boards may or may not have to be designed. If possible, the new types of sensors will be designed to interface to existing signal conditioning, conversion electronics, and communication functions. If not, then a new type of DAS module would have to developed. On the new module a new DAS ASIC would be developed that would provide for all of the signal conditioning necessary to interface the new sensors to a DAS ASIC.

Chapter 3 Requirements

The baseline for the communication controller board calls for 12 internal communication ports. Each of these ports transfers data a rate of 1 Mbps. This creates an aggregate data rate of 14 Mbps of data. The external communication path must be able to handle the 14 Mbps rate, so 16 Mbps data rate was chosen. There is no retransmission capability so any garbled transmissions mean lost data. The sensor sequence numbers should make it obvious which data is missing.

The communication channels are arranged in a star configuration. The phase I communication controller board allows for up to twelve channels (communication pairs) to be connected directly to the communication controller board. Any additional channels require another communication controller board to be installed within the model.

The internal communication link transmits data and commands as single words. Each word is 24 bits long, begins with three start bits, and ends with a parity bit. The parity bit is used for error checking. The system uses even parity. Even parity means that, an even number of high bits are sent for all normal transmissions. If an odd number of high bits are received, the parity-checking block flags that an error has occurred. The data is transmitted MSB first, and LSB last.

Commands sent across the internal communication links are answered by an ACK command. Any commands not answered can cause a timer to run out and trigger a re-transmission of the original command. Data transmissions are not answered by ACK commands. Any data transmissions with garbled start bits (101) is ignored. See figures 2 and 3 for the internal command bit structure.

The external communication link transmits data and commands as single word. Each word will be 28 bits long and begins with three start bits and end with a parity bit. The parity bit is used for error checking. The external system also uses even parity. The data is transmitted MSB first, and LSB last.

Commands sent across the external communication links are answered by an ACK command. Any commands not answered can cause a timer to run out and trigger a re-transmission of the original command. Data transmissions are not answered by ACK commands. Any data transmissions with garbled start bits (101) are ignored. See figures 6 and 7 for the external command bit structure.

The boards to be delivered will have to withstand ± 5g accelerations. There is a shock and a constant vibrational loading requirement.

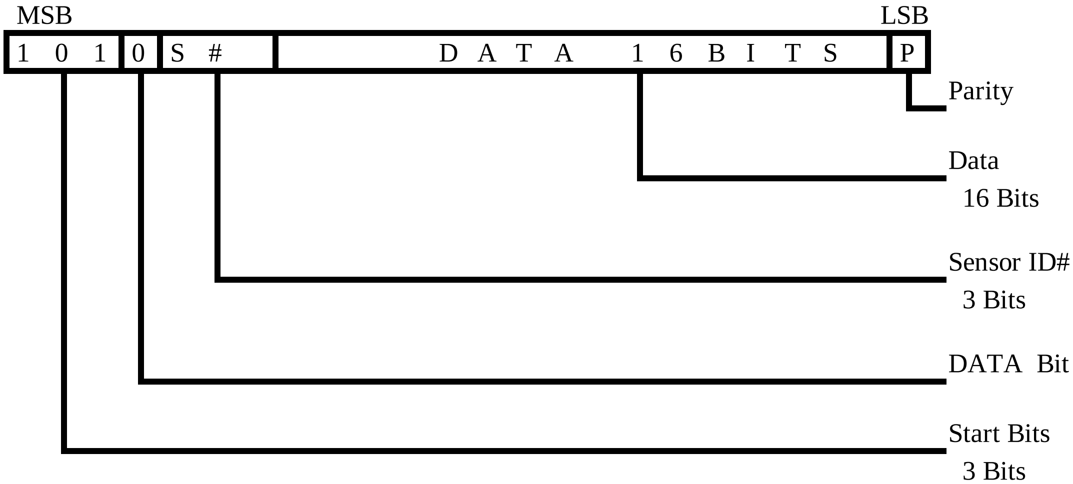


Figure 2 Internal Communication Generic Data Word

Generic Internal Data Word

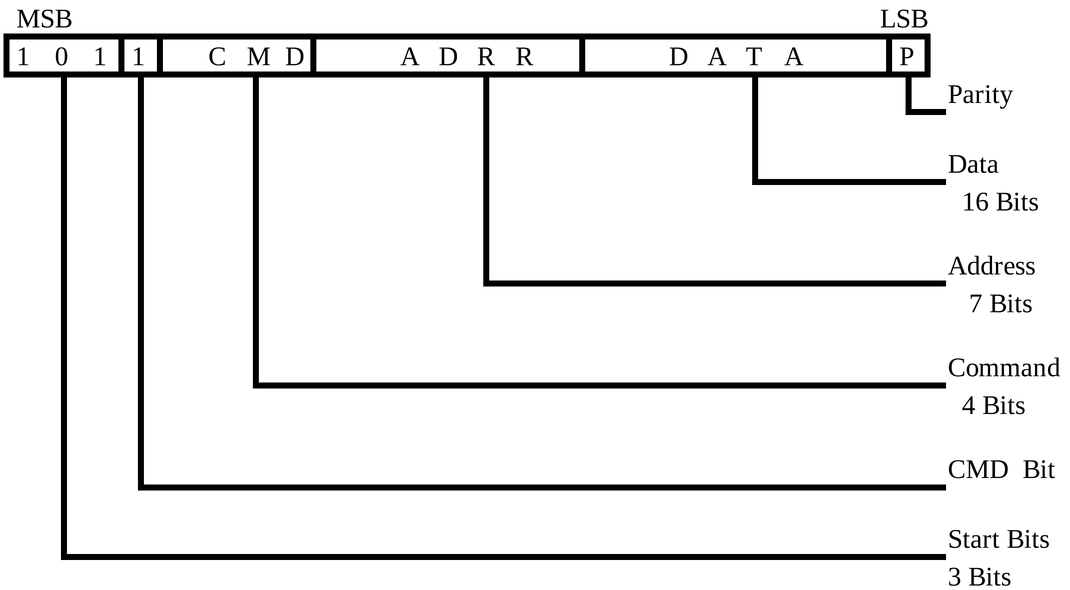


Figure 3 Internal Communication Generic Command Word

Generic Internal Command Word

Note: Transmissions Send MSB First!!!

Chapter 4 Theory of Operation

The communication Controller board is essentially a communications board. In short, its duties involve moving data and commands to and from the external communications link, to the internal sensor board communication links. Commands are generated by the host computer, sent across the external link where they are routed to the appropriate sensor board across an internal communication link. Data comes from the sensor boards and is routed through the communication Controller board to the external link. What follows is a more detailed description of the communication Controller boards operation.

4.1 Initialization

After the configuration of the communication controller board is completed, the communication controller begins setup of the DAS clusters across the internal serial communication by sending a READ command, to read the STATUS register of each of the twelve links. (See figures 2 and 3 for the generic command and data words.) The responses from all twelve links are sent to the host computer for analysis. Any active DAS clusters send back the contents of the Status register, which is used to indicate how many clusters are operational. See table 3 for a list of the cluster ID numbers. After each of the DAS clusters has been polled, the communication controller board sends the data to the host computer using the external communication link, indicating that the initialization sequence has completed. Any additional configuration or calibration sequences are to be handled by the host computer.

4.2 Normal Operation

After successful initialization, the system enters the normal operation mode, when the communication controller sends out the ACQ (Data Acquisition Mode Command) to all of the initialized DAS modules. Each DAS module responds with an ACK (Acknowledge Command). See table 2 for a list of command responses. In this mode the sensor boards continuously acquire data and periodically send the data to the communication controller board. The communication controller collects the data and places it in the FIFO buffer for the external serial link. The communication controller transmits the FIFO buffer using the external communication link to the host computer for storage and display. This sequence is initiated by an ACQ (Start Acquiring Data) command sent from the communication controller to each sensor board. Each cluster prepares, and buffers the data then waits to transmit the data to the communication controller board. Sending a STOP command terminates the Normal mode of data acquisition.

4.3 Synchronization

Periodically it is necessary for the communication controller to re-send an ACQ command to align all of the sensor clusters to proper synchronization. Upon receipt of an ACQ command each DAS module clears its frame counter and enters the normal data acquisition mode. If a DAS cluster is already in the normal mode when an ACQ command is received, it clears its frame counter, and continues taking data.

4.4 Calibration Mode

The calibration mode is entered when the host computer sends a CAL (calibration) command to a DAS module. The module responds by sending back an ACK word. The DAS module then executes the calibration sequence.

4.5 Utility Commands

Commands such as the READ, WRITE, and RESET are utility commands. The WRITE and RESET commands do not require any response other than the ACK command. The READ command initiates an ACK response and then a RESPONSE command is sent back with the contents of the register to be read in the data section.

Command	**Function**	**Bit Pattern**
ACK	Acknowledge the receipt of a command	0000
ACQ	Start Acquiring Data, enter normal mode from calibration mode	0001
CAL	Calibration Command	0010
STOP	Stop Acquiring Data/ Stop Cal Mode Command	0011
READ	Read an memory location	0100
RESET	Reset controller or DAS module	0101
RESPONSE	Data returned word	0110
WRITE	Write to a memory location	0111
	Reserved	1000
	Reserved	1001
	Reserved	1010
	Reserved	1011
	Reserved	1100
	Reserved	1101
	Reserved	1111

Table 1 Communication Controller Board Command List

Command	**Response**	**Notes**
ACK	No Response	
ACQ	ACK	Data is sent asynchronously!
CAL	ACK	
STOP	ACK	
READ	ACK and a RESPONSE command Word	RESPONSE is sent asynchronously!
RESET	ACK	
RESPONSE	ACK	
WRITE	ACK	

Table 2 Command Response Table

Item	Function	Address
None	Deselect for ALL!	0000
CLUS_1	Cluster ID #1	0001
CLUS_2	Cluster ID #2	0010
CLUS_3	Cluster ID #3	0011
CLUS_4	Cluster ID #4	0100
CLUS_5	Cluster ID #5	0101
CLUS_6	Cluster ID #6	0110
CLUS_7	Cluster ID #7	0111
CLUS_8	Cluster ID #8	1000
CLUS_9	Cluster ID #9	1001
CLUS_10	Cluster ID #10	1010
CLUS_11	Cluster ID #11	1011
CLUS_12	Cluster ID #12	1100
Comm	Communication Controller Board	1101
Parallel	Parallel Interface	1110
All	Select All Clusters	1111

Table 3 Cluster ID numbers

Item	**Function**	**Address**
None	Deselect for ALL!	0000 XXX
ISIO_1	Select Internal SIO #1	0001 XXX
ISIO_2	Select Internal SIO #2	0010 XXX
ISIO_3	Select Internal SIO #3	0011 XXX
ISIO_4	Select Internal SIO #4	0100 XXX
ISIO_5	Select Internal SIO #5	0101 XXX
ISIO_6	Select Internal SIO #6	0110 XXX
ISIO_7	Select Internal SIO #7	0111 XXX
ISIO_8	Select Internal SIO #8	1000 XXX
ISIO_9	Select Internal SIO #9	1001 XXX
ISIO_10	Select Internal SIO #10	1010 XXX
ISIO_11	Select Internal SIO #11	1011 XXX
ISIO_12	Select Internal SIO #12	1100 XXX
XSIO	Select External SIO	1101 XXX
	Reserved	1110 XXX
All_ISIO	Select All Internal SIOs	1111 XXX

Table 4 Communication controller Memory Map Upper 4 bits

SIO Register	Function	Address	Data
TX/RX	Reads from RX register, Writes to TX register	XXXX 000	8 Bit Data
Timer Lo	Timer Low byte value	XXXX 001	8 Bit Timer Value
Timer Hi	Timer High byte value	XXXX 010	8 Bit Timer Value
Config	Configuration Register	XXXX 011	8 Bit Value
	Reserved	XXXX 100	None
	Reserved	XXXX 101	None
	Reserved	XXXX 110	None
	Reserved	XXXX 111	None

Table 5 Communication controller Registers Lower 3 address bits

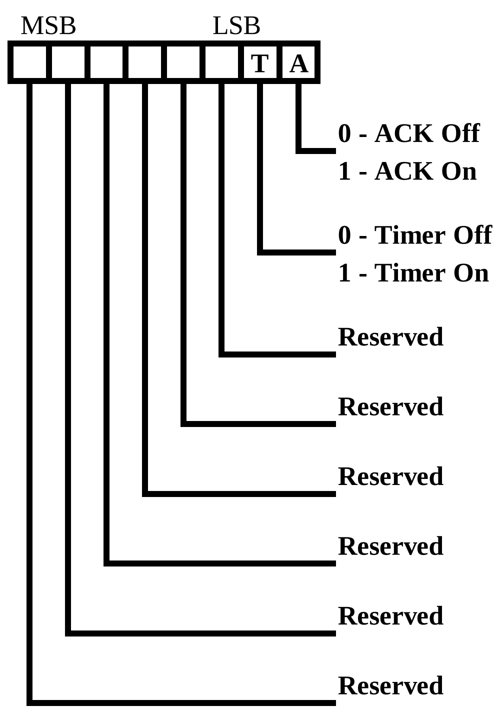


Figure 4 Configuration Register

4.6 Internal Serial Data Rates

The system is based on a maximum transmission bit rate of 1 Mbps per internal link. The number of sensors and the sample rate may vary on a single link, as long as the 1 Mbps transmission rate is observed. The data rate is 666.67 KBPS. Notice that the word size is fixed at 24 bits and the data is 16 bit values for each of the sensors, as well as the A/D. The actual number of bits varies according to the sample rate. Table 6 below, is an example of how the system could work.

Sensor	**Sample Rate**	**# of Bits in Word**	**bps**
Shear Stress #1	20 kHz	24	480,000
Shear Stress #2	20 kHz	24	480,000
Pressure	250 Hz	24	6,000
AoA1	250 Hz	24	6,000
Temperature	250 Hz	24	6,000
Engineering #1	250 Hz	24	6,000
Engineering #2	250 Hz	24	6,000
Engineering #3	250 Hz	24	6,000
		Total	996,000

Table 6 Data Rate Example 1

The numbers used are not the actual sample rates, but should be in the ballpark. Notice that the communication overhead is in the number of bits per word. There are 16 bits of data for a word of 24 bits. The 666.67 kbps number comes from the maximum number of data bits that can be transferred a single link in one second.

Table 7 is another example showing how changing the sensors would affect the sample rates of the system. In this second example, the sample rate for 7 sensors can be as high as 3 kHz while there is only one shear stress sensor in the cluster. Remember the system can handle both of these clusters, the clusters do not have to be the same.

Sensor	**Sample Rate**	**# of Bits in Word**	**bps**
Shear Stress #1	20 kHz	24	480,000
Pressure	3 kHz	24	72,000
AoA1	3 kHz	24	72,000
Temperature	3 kHz	24	72,000
Engineering #1	3 kHz	24	72,000
Engineering #2	3 kHz	24	72,000
Engineering #3	3 kHz	24	72,000
Engineering #4	3 kHz	24	480,000
		Total	984,000

Table 7 Data Rate Example 2

4.7 Power ON Reset Sequence

1) Communication Controller configures itself after a power on reset.
2) Communication Controller broadcasts a read status command to all DAS units.
3) Communication Controller transmits to the host a 0B0055h command.
4) Communication Controller sends the host the status registers of all of the DAS units that respond.
5) The Host sends a series of write commands to set the Timer for each SIO/DAS.

6) The Host sends write command to set the Config register of each SIO/DAS.

7) Repeat steps 5 and 6 for each SIO/DAS and for the XSIO also.

8) The Host sends a CAL command to a DAS unit.

9) The Host will send any additional commands for the specific DAS unit.

10) The Host sends a Stop command to exit the calibration mode.

11) Repeat steps 8 through 10 for all DAS units.

12) If necessary, the Host will send a broadcast ACQ command to all of the DAS units to begin taking Data.

*Host commands

Power ON Reset Sequence commands and example data.

Step #	Command	Operation	Example	Notes
5	VB8VXX	Write to Timer Lo	1B8955	55h to SIO#1 timer Lo
6	VB8VXX	Write to Timer Hi	1B8A44	44h to SIO#1 timer Hi
8	FBFB0X	Write to all Configs	FBFB03	ACK and Timer Bits ON
9				
10				

V – SIO identifier
X – any value

*Commands as sent by the host without start or parity bits.

Chapter 5 Prototype Board Design

5.1 Prototype Communication Controller Board

The first phase of the project was the design and development of the prototype communication Controller board. The second phase will be the implementation of the communication Controller board as an ASIC. The prototype board contains: twelve pairs of serial drivers/receivers for the internal communication links, RAM, EEPROM, a crystal for synchronizing the timing of the state machines, and the discrete parts that will make up the communication Controller ASIC. The prototype communication Controller board contains a regulator for power conditioning. This reduces the incoming +8V to +5V, for the rest of the electronics on the board. All of these parts are in the communication Controller board design. See figure 5 for a block diagram of the communication Controller board.

Communication Controller Block Diagram

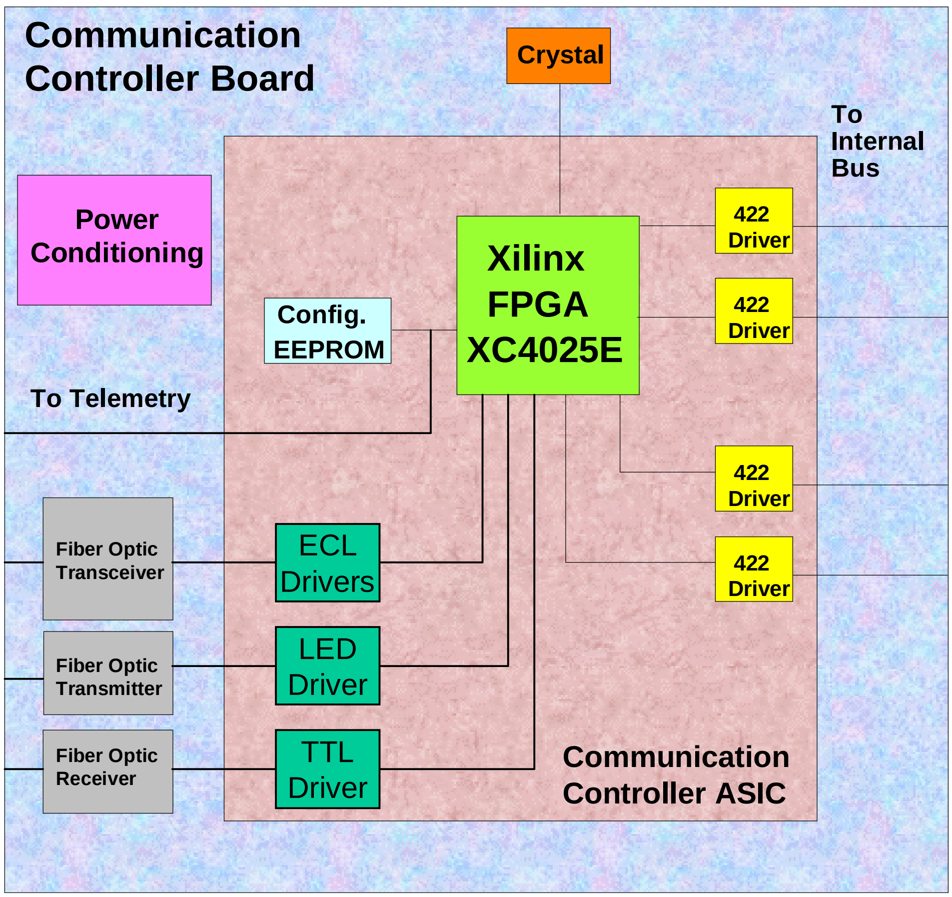


Figure 5 Communication Controller Board Block Diagram

The following parts comprise the portion of the communication Controller prototype design that makes up the ASIC design: FPGA, differential drivers, EEPROM. In figure 5, the ASIC portion is placed within the red block. The FPGA chip houses the state machines that manage the communication Controller board. The initialization code is placed in an EEPROM. The EEPROM was chosen for the ability to be reconfigured while still maintaining non-volatility.

The communication Controller board uses a Xilinx XC4025E FPGA chip. This chip has 32k bits of RAM internal. The basic building unit is a Configurable Logic Block (CLB) and the 4025 has 1024 CLBs and 256 IO blocks. The 4025 also has 2560 flip flops specified along with 25,000 equivalent gates. The FPGA contains a boot loader that configures the device from either a serial or a parallel external memory. In many applications, the memory is a ROM, for this design an EEPROM is used as non-volatile storage of the configuration data. The Xilinx chip is configured using the mode pins M0, M1, and M2, to place the chip in either the master parallel mode, or the master serial mode. A jumper allows the user to select either parallel or serial mode. When the serial mode is used, a four-pin header provides the serial interface. In the parallel mode, a 40-pin header provides access for all of the data and address lines.

The communication Controller board contains a test header for accessing the EEPROM and disabling the FPGA. This test header is used for initial testing and configuration of the FPGA via the EEPROM. Test headers are also available for the twelve pairs of serial communication lines.

The memory of the communication Controller board is arranged in a linear address space. The internal data registers of the communication Controller FPGA use the lowest address of the address space. See tables 4 and 5 for a memory map of the communication controller board's address space. Note that the memory uses a standard parallel bus. This bus can be used for future parallel interfaces.

The FPGA is implemented using synthesized VHDL. "VHDL serves as a hardware description language for simulation as well as for synthesis."[1] One reason that VHDL was chosen is because it allows the FPGA design to be retargeted to an ASIC with a minimum of redesign necessary. In the ideal case, the design is re-synthesized using an ASIC standard cell library instead of the FPGA library. In reality, the need will arise for some modifications of the design to meet the constraints of the system.

5.2 Host Telemetry Interface

The host computer is interfaced through an external serial I/O block, or the parallel interface that accesses the onboard RAM. On the communication controller board, the main state machine oversees both external interfaces. The serial interface uses the fiber optic transceivers, while the parallel interface is used for Radio Frequency (RF) telemetry.

The information that is transmitted is made up of single words that originated from the DAS modules. See figures 6 and 7 for a description of the words sent externally from the model using external serial I/O block. Note that the words are sent out as soon as possible with as little buffering as is necessary.

There are two fiber optic transceivers on the communication controller prototype board. The reason for two transceivers is to mitigate risk by placing two different options on the prototype board. The best option will be the one used on the ASIC version of the communication controller. Both of the fiber optic interfaces are overseen by the main state machine. Before data is transmitted to the host computer, the communication Controller removes the parity bit from the words as it receives them from the DAS modules. The communication Controller adds the source address to the beginning of the word and stores the word in the FIFO buffer memory. The communication Controller then removes the word from the buffer for transmission to the host computer. The data words are transferred to the host computer as quickly as is possible for the system. It is the host computer's responsibility to time tag the data as it arrives. Each sensor cluster places a sensor ID in the data block. The sensor ID is an indication that the data has been updated, because not all of sensors update in a serial fashion.

The host computer is also responsible for setting up the scan sequence. Note that the scan sequence must never hit the same sensor twice in a row, because it will destroy the capability of detecting missing words! Ping ponging between two sensors alleviates this problem.

The calibration mode of operation is under the control of the host computer. This means that on power up, it is the host computer that must send a calibration command. The host computer can always change the mode to calibration, should the host computer decide to do so. Since the host computer can monitor the conditions of the electronics, by looking at the engineering data, it can decide when conditions warrant re-calibration.

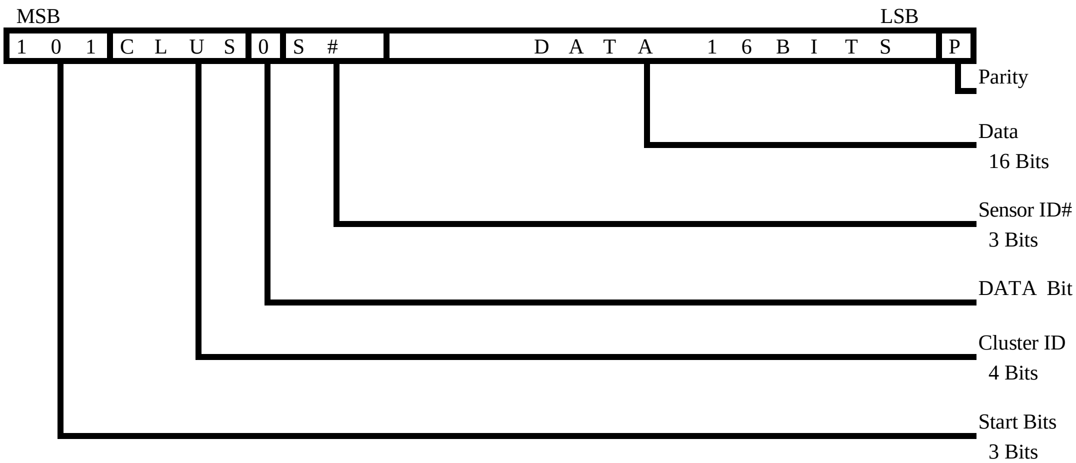


Figure 6 External Communication Generic Data Word

Generic External Data Word

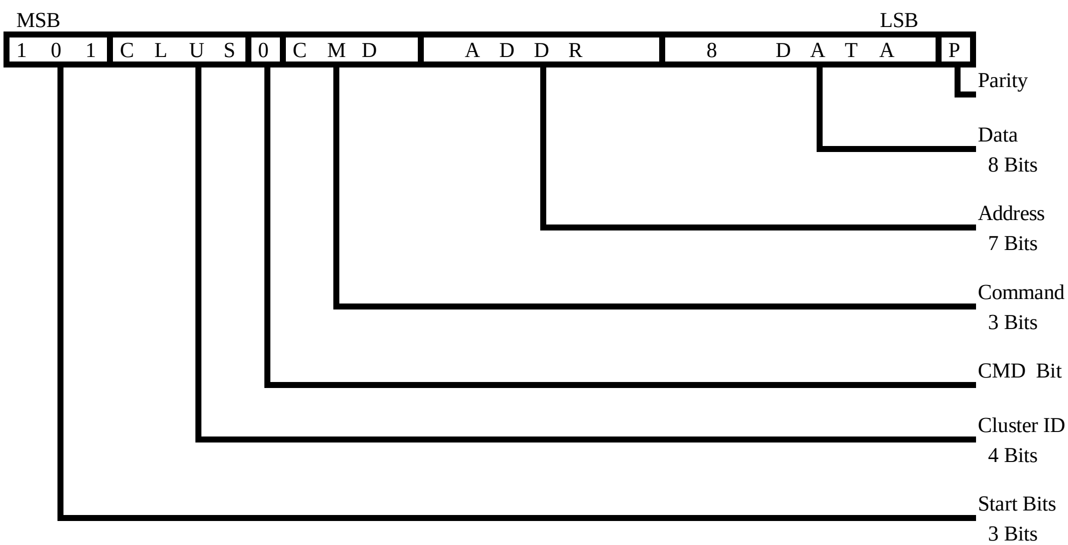


Figure 7 External Communication Generic Command Word

Generic External Command Word

Note: Transmissions Send MSB First!

Chapter 6 FPGA Design

The FPGA was designed using a top down design methodology and was implemented using VHDL. "The experience, researchers, software developers, and other users with VHDL since it became the IEEE standard in 1987 indicates this language is sufficiently rich for designing and describing today's digital systems."[2] There is one FPGA design on the communication Controller board, and it is called the Communication Controller FPGA.

The VHDL for the Communication Controller FPGA was simulated using ORCAD Express for Windows Simulator. The simulation was performed for functional verification of the FPGA using a test bench. "A test bench is a model that is used to exercise and verify the correctness of a hardware model."[3] When possible backannotated timing simulations were performed after the VHDL was been synthesized, placed, and routed. Exemplar Logic's Leonardo tool was used for synthesis and Xilinx Design Manager was used for place and route.

For the design of the FPGA the following "Best Practices" were used wherever possible:

- All VHDL will adhere to the IEEE 1164 standard.
- Double buffer all asynchronous inputs to the state machine to eliminate any metastability issues.
- Both edges of the clock will be used.
- Clock the state machine on one edge of the clock and clock the registers on the other edge of the clock.

- All registers will be synchronous with synchronous resets and loads.

6.1 Communication Controller FPGA

"An FPGA is a general-purpose, multi-level programmable logic device that is customized in package by the end users. FPGAs are composed of blocks of logic connected with programmable interconnect. The programmable interconnect between blocks allows users to implement multi-level logic, removing many of the size limitations of the PLD-derived two-level logic structure. This extensible architecture can currently support thousands of gates of logic at system speeds in the tens of mega-hertz."[4]

The Communication Controller FPGA design consists of the following: the main state machine, the twelve internal serial I/O blocks, and the external serial I/O block, FIFO, Bus interface, and Clock block. See figure 8 for a block diagram of the FPGA's internal architecture. The design is driven by the main state machine, which controls all aspects of the blocks internal to the FPGA: serial to parallel conversions, FIFO buffer memory, parallel to serial conversions, and external serial I/O functions. "VHDL provides convenient constructs to describe various forms of state machines at various levels of abstraction."[5]

Incoming data will arrive at the internal serial I/O block. The data will be converted from serial to parallel and then stored in the internal buffer memory (FIFO). The data will then be moved from the buffer memory out to the external serial I/O block. Commands sent from the host computer (external serial I/O) will be converted from serial to parallel and passed to the appropriate SIO block for conversion from parallel to serial, and transmission to a DAS module.

FPGA Internal Block Diagram

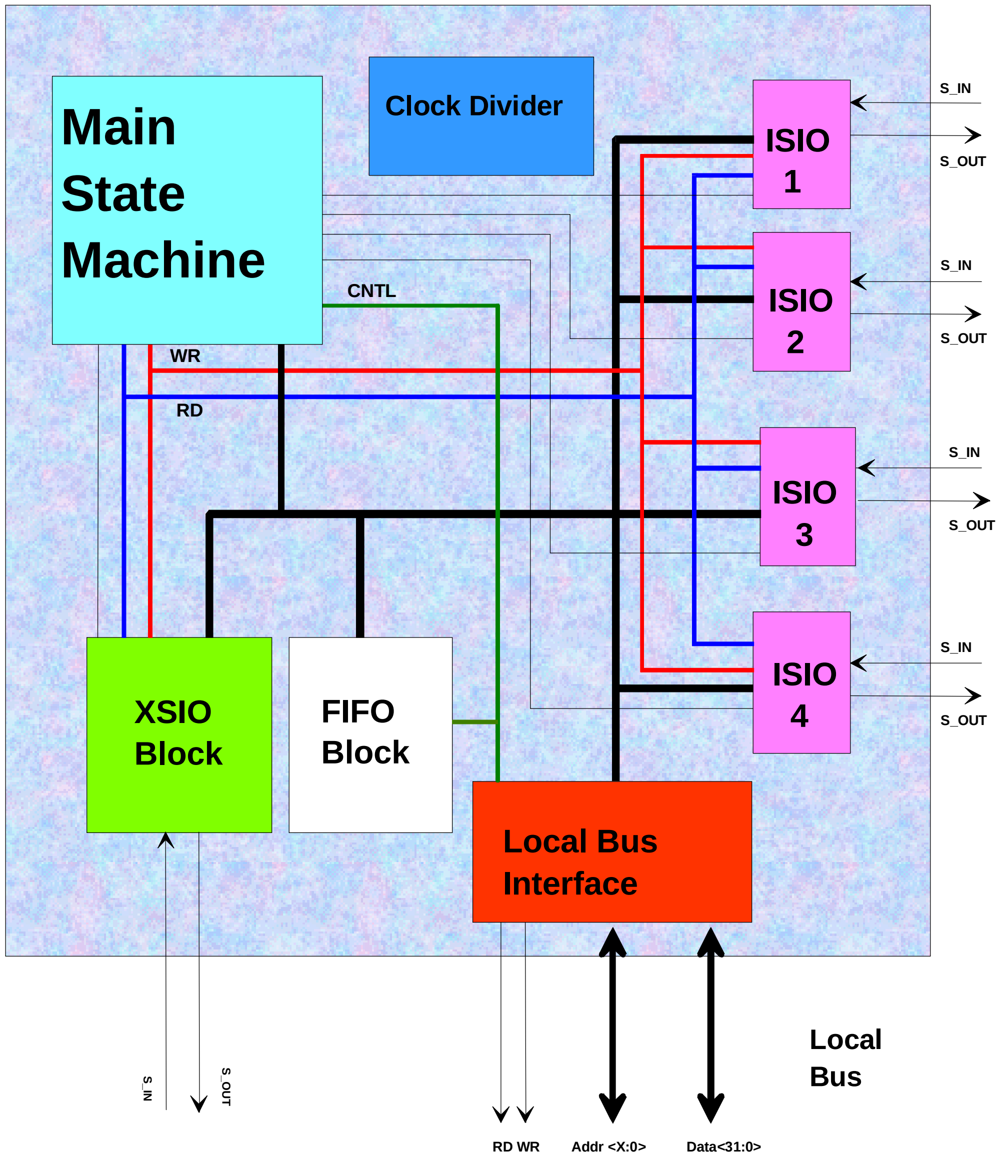


Figure 8 Communication Controller FPGA Block Diagram

6.2 Main State Machine

The main state machine allows operators to read and write to all of the registers. The foremost task is to move data from the internal serial I/O blocks to and from the buffer memory and then to and from the external serial I/O communication ports. See figure 9 for the state transition diagram for the main state machine. The state machine checks each of the receivers SIO's Word_Ready signals in a prioritized fashion starting with the external communication FIFO. The priority for checking the FIFO is first the external communications, then the SIO #1, then the SIO #2, and so on, until SIO #12 has been checked. To check the receiver's FIFO the state machine looks at the Word_Ready signal line from the SIOs or XSIOs. If the signal is low, then data has not been received and the state machine proceeds to check each of the other SIOs. If, however, the signal is high, then the data must be moved to the FIFO to be acted upon. First, the destination address is checked to see if the communication Controller module is the destination. If the destination is not the communication Controller, then data is moved into the transmitter register of the SIO indicated by the destination address. If the destination is the communication Controller, then the next step is to check the command/data bit. If this bit is high, then a command has been received. The command is parsed to determine which command sequence should be executed.

See Table 1 for a list of all possible commands for the communication Controller module. There are three types of commands for the communication Controller to execute. Read commands, which require the communication Controller to send data to the external communication device. Write commands require the communication Controller to replace data with new data. There are also commands that have no data associated with them, such as the Reset FPGA command.

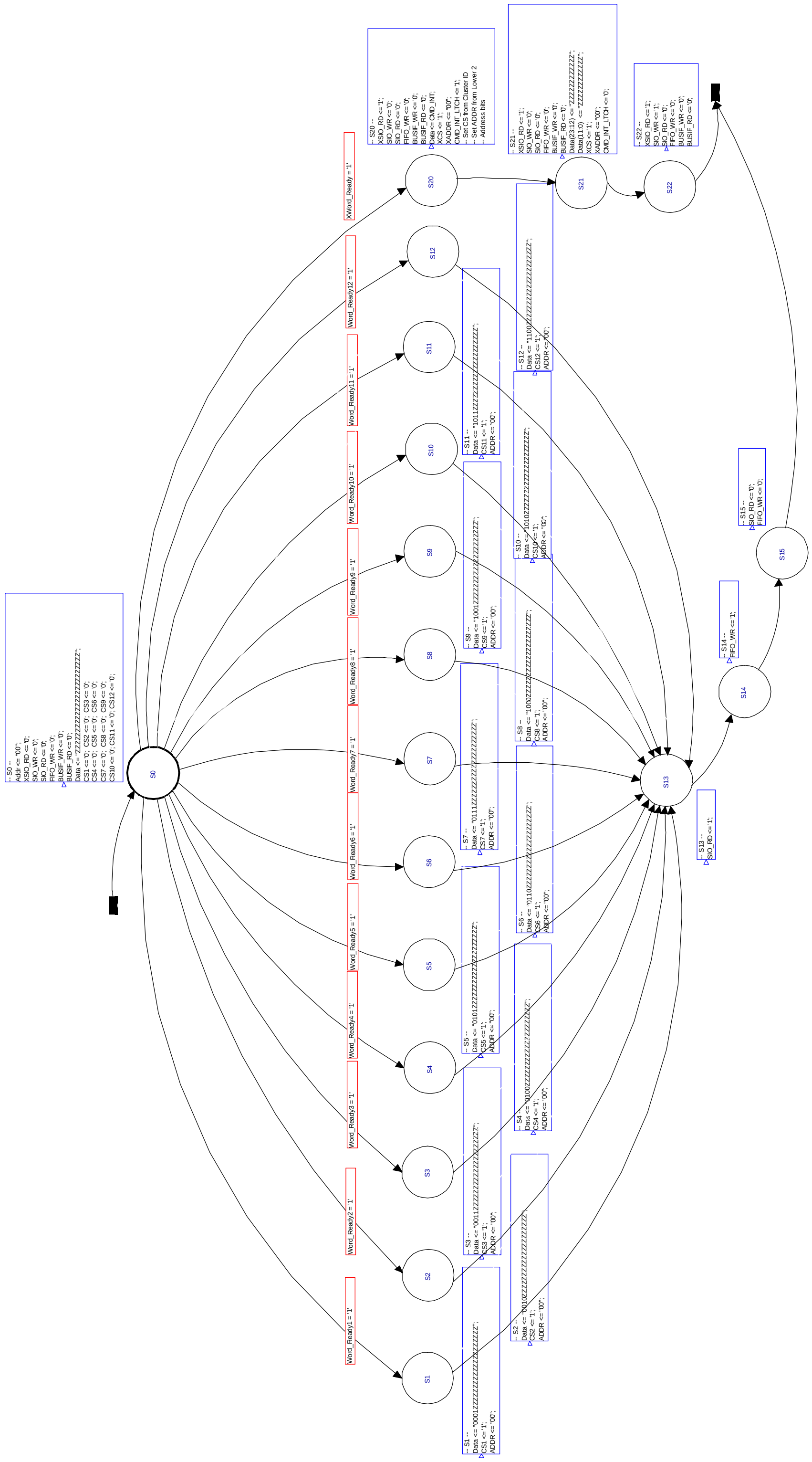


Figure 9 Main State Machine State Diagram

6.3 Internal Serial Input/Output

The serial I/O block is where the bulk of the communication processes occur. See figure 10 for a block diagram of the internal serial I/O portion of the FPGA. The state machine within the Decode block controls the serial I/O block. This state machine off loads I/O processing from the main state machine, thus allowing the multiple serial I/O sections to operate in parallel without overloading the main state machine.

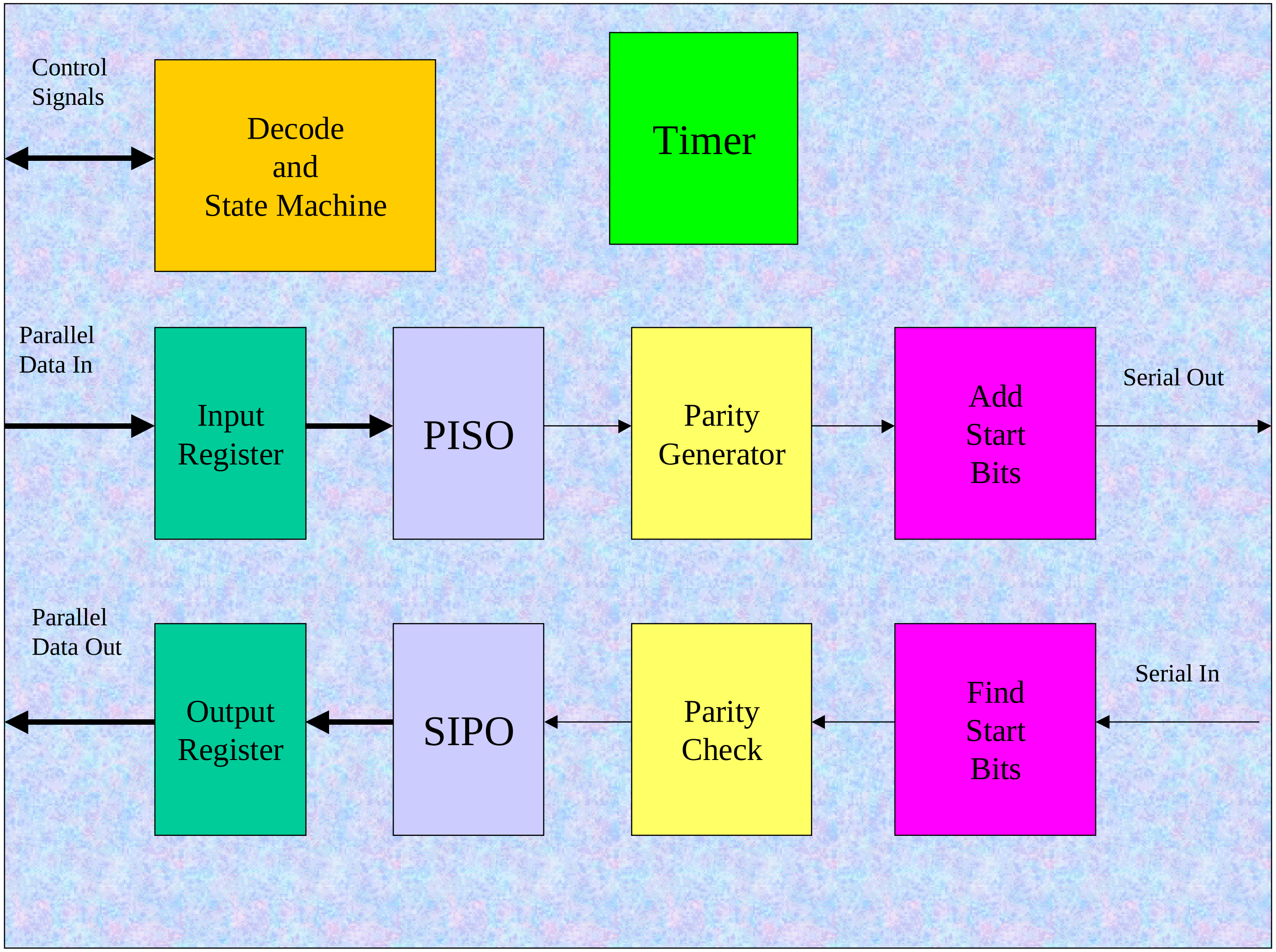


Figure 10 Internal Serial I/O Block Diagram

The serial I/O block is broken into the two data flow paths, one path for incoming data and one for outgoing data. In the outgoing data path, first the main state machine first checks to see if data is waiting to be transmitted and if the transmitter is busy. If the transmitter is not busy then data is transferred from the output buffer called the SIO_OUT_REG, to the output shift register. The data is then shifted out one bit per clock cycle, using a 1 MHz clock. When the last bit of data has been shifted out of the shift register, the Parity generation block sends the parity bit to the ADD_START block. Here the three start bits are prepended and the parity bit is appended. The serial stream leaving the ADD_START block is sent to the transmitter chip where the signal is converted to differential signals to be sent across the wires.

The second path begins with data being received at the differential receiver chip. The Find_Start block monitors the incoming signal lines for the three start bits. If the start bits are received then data is allowed to pass to the Parity check block and the Shift register. The data bits are shifted in one bit per clock cycle until the serial to parallel shift register is full. Once the register is full the 20 bits of data and parity bit, less the three start bits, are transferred to the shift register, and the parity check block. As each bit is received, it is passed through the parity check block. When the last bit is passed through the parity check block the Parity_Check signal will indicate whether the word just transmitted passed the parity check. If the word passes, the parity check Word_Rdy signal is sent to the main state machine so that the data can be transferred to the FIFO. Once in the

FIFO buffer, the data will wait to be transferred to the external serial I/O block for transmission off the model to the data acquisition host computer.

For communication internal to the model, the communication links use a differential serial scheme. This allows for communications across only 4 wires per channel. The data is sent across as 24 bits with three start bits and no stop bits and one parity bit. The data rate is 1 Mbps for the internal communication links. The serial link transfers data as a single word. See figures 2 and 3, for a general description of the internal command or data words.

Each DAS has its own cluster ID. Cluster number 0 is not used to select anything. The DAS clusters use cluster 1 through 12. Cluster 13 is used for communication controller board. Cluster 14 is used for the parallel interface. Cluster 15 is used for broadcasting commands to all of the internal serial links at the same time.

The command/data bit received will signify whether the rest of the block is a command or data. The internal serial communication links use even parity for error detection. This means that for valid transmissions, the word will have an even number of high bits including the parity bit. If a word is received with an odd number of high bits, an error has occurred and the word is ignored.

On the transmit side, the parity bit is generated as each word is transferred out of the SIO_OUT_SR output shift register. A parity bit is appended to the end of the data word as it is transferred out of the SIO shift register. On the receive side, a similar process takes place. As the words are received, they are run through the parity generation algorithm. When the transmitted parity bit is received, it is used to generate the check parity bit. If the parity check bit is high, then the transmission was received without any errors and the word is used. An ACK command is sent to the source to acknowledge the receipt of a valid word (If the ACK bit is set high in the Config Register of the SIO block). However, if the parity check bit does not match the word is thrown away.

If the timer bit of the configuration register is turned on then the communication controller will automatically retransmit commands if an acknowledge is not received and the timer counts down to zero. The timer values need to be set to about 4 times the length of transmission. The large value will allow for receipt of words that are transmitted before the ACK command.

6.3.1 Internal Serial I/O Decode State Machine

The serial input and output functions are controlled by the Decode State machine. This state machine has 14 states. See figure 11 for the state diagram. This state machine governs the operations for all of the serial blocks. State 13 is the reset state where all of the outputs are set to inactive states. If at any time the Reset line goes high, then on the next rising edge the state machine will go to State 13. The state machine will go to state 0 after one clock cycle. The state machine will stay in State 0 until one of the following events occurs: the Data_Rdy goes high, the Data_Waiting is high and the TX_Busy is low, Timer_Flag goes high, or the RD line and the CS go high.

If the state machine is in state 0 and the Data_Rdy signal goes high, then the next state will be state 1. In State 1 the PC_Latch signal is examined. If the PC_Latch signal is low, then the data will be ignored and the next state will be state 0 since the data received did not have the proper parity. Another function of state 1 is to latch the CMD portion of the incoming data into CMD_INC. If the PC_Latch signal is high, then the next state will be state 2. In state 2 the CMD_INC register is checked to see if it matches an acknowledge command "10000". If an acknowledge command was received then no other action is necessary and the next state will be state 0. If the data received was not an acknowledge command then the next state will be state 3. In state 3 the W_Rdy signal is raised to indicate that a valid word has been received, and bit 0 of the configuration register is checked. If the Config register bit 0 is not set high then the current configuration does not reply to commands with an ACK command, and the next state is state 0. If the Config register bit 0 is set high an ACK command must be sent to the DAS, so the next state will be state 4 (See figure 4 for the bit patterns for the configuration register). In state 4 the TX_Busy signal is checked. If TX_Busy is high then the transmitter is busy so the state machine will stay in state 4 until the TX_Busy line falls. Once the TX_Busy line is detected as a low, then the next state will be state 5. In state 5 the Load signal is brought high while the ACK signal is also held high. This will cause the ACK command in the command register within the SIO_OUT_REG block to be transferred to the shift register in the SIO_OUT_SR block. This causes the command to be transmitted to the DAS module. After one clock cycle, the state will be changed to state 6.

State 6 is a where the state machine will stay until the Data_Rdy signal falls low. This will prevent the state machine from retriggering on the Data_Rdy signal and going through the whole process again. Once the Data_Rdy signal has fallen, the state machine will return to state 0.

If the Data_Waiting signal goes high and the TX_Busy signal is low then the state machine will change to state 11. In state 11 the TX_Busy signal is checked again. If TX_Busy is high then the transmitter is busy so the state machine will stay in state 11 until the TX_Busy line falls. Once the TX_Busy line is detected as a low, then the next state will be state 12. In state 12 the Load signal is brought high while the ACK signal is held low. This will cause the data in the data register within the SIO_OUT_REG block to be transferred to the shift register in the SIO_OUT_SR block. This causes the data or command to be transmitted to the DAS module. . After one clock cycle, the state will return to state 0.

If the Timer_Flag goes high then on the next rising clock edge the state will go to state 7. State 7 checks bit 1 of the Config register. This bit turns on and off the retransmission of commands on a time out. If bit 1 of the config register is zero then the re-transmission is turned off and the next state is state 0. If the bit 1 is high then on the next rising clock edge, the state will be state 8. In state 8 the TX_Busy signal is checked. If TX_Busy is high then the transmitter is busy so the state machine will stay in state 8 until the TX_Busy line falls. Once the TX_Busy line is detected as a low, then the next state will be state 9. In state 9 the Load signal is brought high while the ACK signal is held low. This will cause the command in the data register within the SIO_OUT_REG block to be transferred to the shift register in the SIO_OUT_SR block. This causes the command to be retransmitted to the DAS module. Notice that unless the DAS module sends an ACK command to the communication controller as an indication of having received the command, the communication controller will continue to retransmit the command every time the timer counts down to zero. On the next clock, the state will be state 10. In state 10 the Load_Data signal is brought low. If the Timer_Flag signal falls then the next state will be state 0, otherwise the state will remain state 10.

If while in state 0 the RD signal goes high and the CS signal is also high, then the next state is state 12. In state 12 the W_Rdy signal is cleared. The next state is state 0.

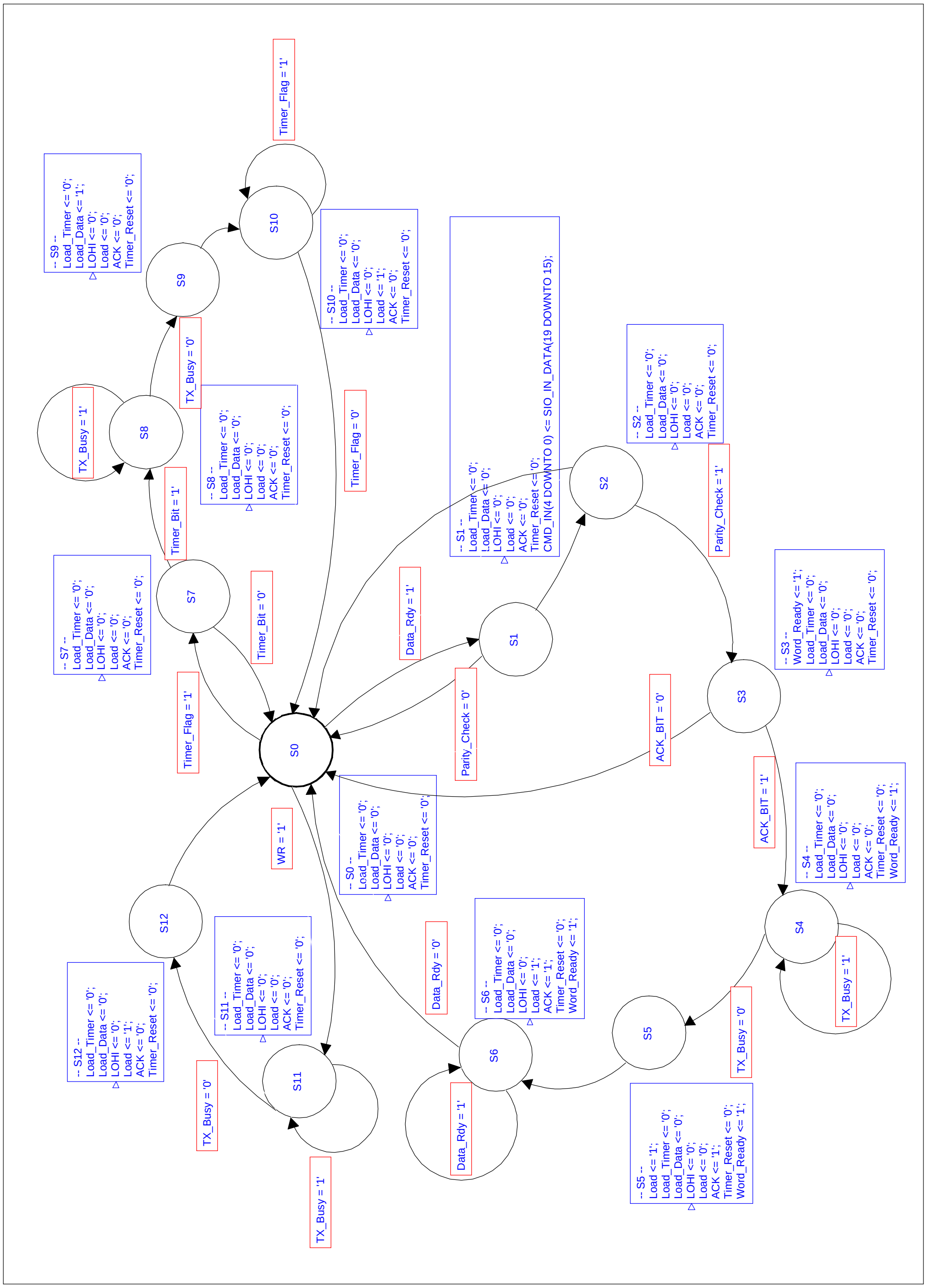


Figure 11   Internal SIO Decode State Diagram

6.3.2 Internal Serial I/O Timer

The timer block is used to make sure that the DAS module receives commands. The DAS module will send an ACK command if it receives a valid command. The ACK command is used by the SIO block as an indication of receipt of a command, therefore if the timer counts down to zero, the command was not received and should be retransmitted. The timer block does this by using a 16-bit down counter. The counter is loaded with a count value that represents the time greater than the transmission latency inherent in the system. The timer uses a 1 MHz oscillator, so each count represents 1 us.

The 16-bit timer register is accessed as two 8-bit registers. The LOHI signal is used to select which registers can be written to, and which register is allowed to output is value on the Reg_Out bus. When the counter hits zero then the Timer_Flag is brought high, this flag will stay high for one clock cycle or 1 us. Also, when the count hits zero, the value in the timer register is automatically loaded into the count register. If the Timer_On signal is high then the counter will decrement on each clock cycle until the counter reaches zero again. Data is written into the timer register, called Reg16, when the Load_Timer signal is high. While the Load_Timer signal is active, the LOHI signal is used to pick which of the bytes in the Reg16 are written to. If LOHI is low then the lower byte of Reg16 is written to, if LOHI is high then the upper byte is written into. The LOHI signal also selects which byte is output on the Reg_Out bus. Again LOHI = 0 selects the lower byte of Reg16, while LOHI = 1 selects the upper byte.

The timer will initially be disabled upon power up, or reset. Before the timer is enabled, it should be loaded with the two timer values. The timer values need to be set to about 4 times the length of transmission. The large value will allow for receipt of words that are transmitted before the ACK command.

6.3.3.0 Internal Serial Transmitter

The transmitter portion of the serial I/O block is comprised of the SIO_OUT_REG, SIO_OUT_SR, Parity, and the ADD_Start blocks. The SIO_OUT_REG register is temporary storage for data to be transmitted, and where the ACK command is stored. When the TX_Busy signal is low, the shift register is not in use and data will be transferred to the shift register. The data will leave the shift register as a serial bit stream to go to both the Parity and the ADD_Start blocks. After the last data bit has been sent to the ADD_Start block, the parity bit will be appended to the serial stream.

6.3.3.1 Internal Serial Transmitter Shift Register

The serial transmitter shift register block is essentially just a state machine and a counter. This state machine has 3 states. State 1 is the reset state where all of the signals are set to zero (S_Data_Out, Busy, Busy_Out, Count and Q). If at any time, the Reset line goes high, then on the next ACLK (1 MHz), rising edge the state machine will go to the reset state. The state machine will stay in the reset state until the load signal goes high. If the load signal is high during a low to high transition of the ACLK, then whatever is on the data lines is latched into the shift register. This is the second state. The shift register is loaded with the high MSB (Most Significant Bit) bit in the bit 19 location and the LSB (Least Significant Bit) in the bit 0 location. If load does not go high, then the state machine will remain in the reset state.

When entering state 2 the busy signal is set high. The busy signal is used externally to indicate that data is being transmitted. Internally the busy signal is used as an enable for the shift register. The busy signal is also used to trigger the state machine to move into the third state.

While the busy signal is high, the shift register will shift data out using the rising edge of the ACLK. Since the counter is enabled by entering state three, and it uses the rising edge of the ACLK it will increment from zero through 24 as the data is being shifted out. When the count reaches 24 (20 data bits plus the 3 start bits and the parity bit) the Busy signal is brought low to indicate that the data has been transmitted. On the next rising edge of the 1MHz clock, the state machine will transition to reset state and the process is ready to go again. For detailed timing information refer to figure 12.

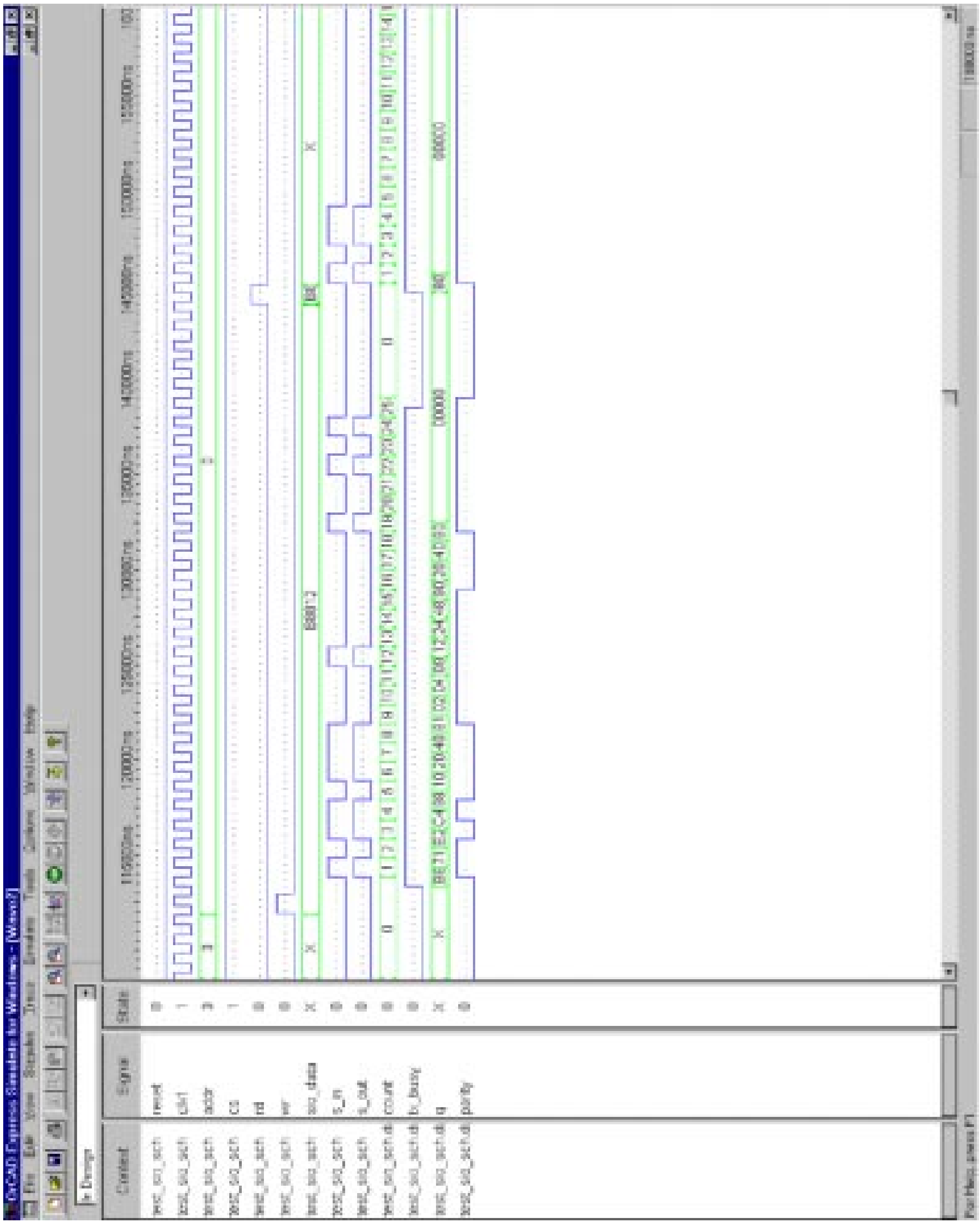


Figure 12   Serial Transmitter Timing Diagram

6.3.3.2 Internal Serial Transmitter Parity generation

The parity generation block generates even parity for the 16 bits of data within a word. See figure 13 for the representative schematic of the parity generation circuit. Although the parity block is implemented in VHDL (see appendix B), a schematic diagram is easier to understand for this simple circuit. As is shown on the schematic only three components need make up the parity generation block. The inverter is used to change the incoming EN signal into a clear signal for the flip-flop. The flip-flop uses the same clock signal that is used for shifting the bits out of the shift register. The most critical part of the circuit is the exclusive or gate. This gate compares the incoming data to the last bit latched into the flip-flop. Since the EN signal will keep the flip-flop cleared when not enabled, the flip-flop will contain a zero when first used for generating a parity bit. As each successive bit arrives it will be xor'd with the output of the flip-flop and latched into the flip-flop. The xor function will output a one only if the incoming data bit or the flip-flop output is a one, but not if both are high. This means that the flip-flop will contain a one whenever an odd number of high bits have been presented to it. If an even number of high bits are clocked in then the output will be zero.

The parity bit is appended to the end of the data stream by the ADD_Start block. Adding the output of the parity generation block to the data word has the effect of insuring that the data plus parity will always have an even number of high bits. It should be noted here, that the start bits do not go through the parity generation block. However, because the start bit pattern "101" has an even number of high bits, it does not change the number of bits necessary for even parity. All words received that do not have a valid parity bit will be ignored.

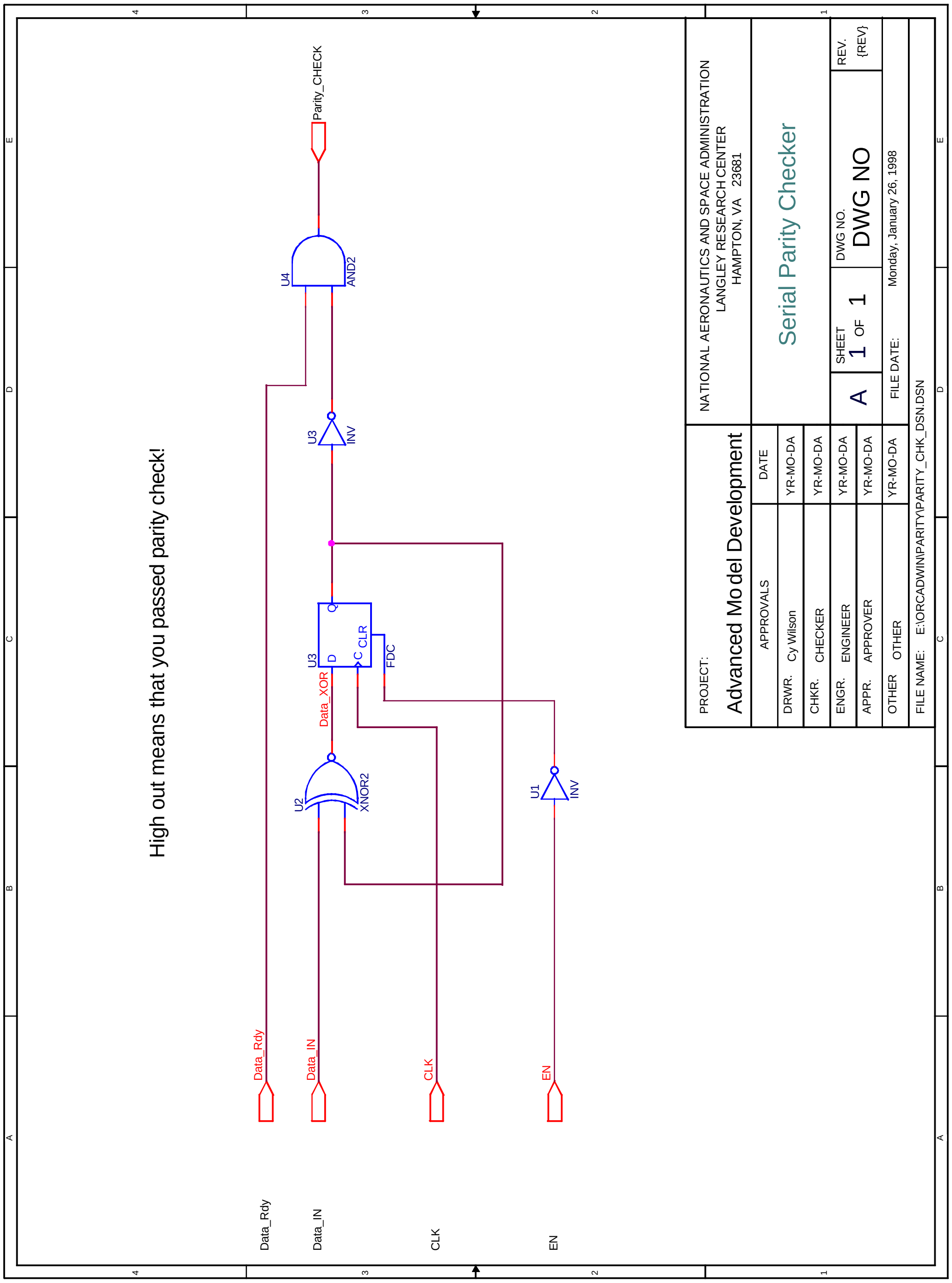


Figure 13 Parity Generation Schematic Diagram

6.3.3.3 Internal Serial Transmitter Add_Start Block

The Add_Start block prepends the three start bits "101" to the serial data stream. At the end of the data word, the parity bit is appended to the word as the last bit in the stream. This creates a 24-bit word to be transferred across the internal serial link. The Add_Start block puts the start bits "101" into the serial bit stream when the EN signal is high. When the start bits have been shifted out the data is then shifted out. When the counter reaches 23 (0-23 for 24 counts) the parity bit is shifted out.

6.3.4.0 Internal Serial Receiver

The receiver portion of the serial I/O block is comprised of the SIO_IN_REG, SIO_IN_SR, Parity_Check, and the Find_Start blocks. The Find_Start block looks at the incoming data for the start bit pattern "101", and when detected the EN signal is brought high. The data will leave the Find_Start block as a serial bit stream to go to both the Parity_Check and the shift register blocks. When the EN signal is high, the shift register will clock in each bit, and the Parity_Check block will check each bit. After the last data bit has been sent from the Find_Start block, the parity check bit will be latched, and the Data_Rdy signal will be brought high to indicate that data has been received. The SIO_IN_REG register is temporary storage place for data to be stored. Data is latched into the SIO_IN_REG when the Data_Rdy signal goes high. The Decode state machine will use the Data_Rdy signal to trigger the W_Rdy signal. When the main state machine sees the W_Rdy signal is high, it will read the data out and act on the data accordingly.

6.3.4.1 Internal Serial Receiver Find_Start Block

This block begins by constantly checking the receive lines for the start bit pattern “101”. When this bit pattern is recognized the 21 bits that follow are passed to the parity checking block and the receiver shift register. In addition to looking for the start bits this block performs a pseudo phase lock loop function. The incoming data is not synchronous to the Find_Start block. Therefore, the incoming data is double buffered to synchronize it to the 1MHz clock. The 1MHz clock is the signal used to clock data into the Parity_Check and SIO_IN_SR blocks. Once the start pattern “101” has been clocked into the Find_Start block the Data_EN_OUT goes high to indicate that data is on the Data_OUT line and can be clocked using the 1MHz clock signal. The Data_EN_OUT will stay high for the 21 clock cycles of 1MHz clock necessary to clock the 20 bits of data and the parity bit into the parity checking block and the receiver shift register. After the parity bit, the Data_EN_OUT signal will fall, and data will not be sent out on the Data_OUT line. See figure 14 for the state diagram for the Find_Start block.

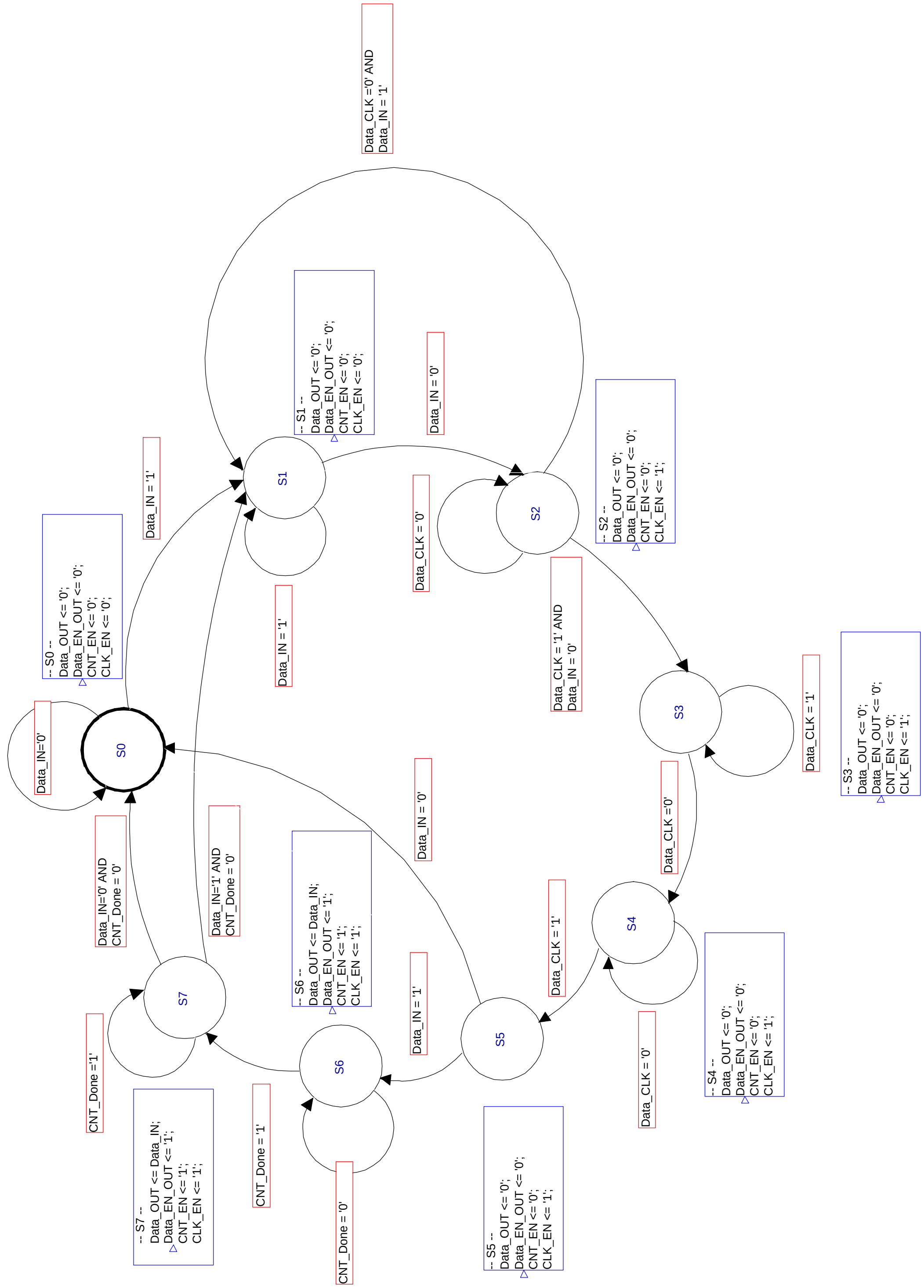


Figure 14 Find_Start State Diagram

6.3.4.2 Internal Serial Receiver Parity Checking

The parity-checking block checks for even parity within the 16 bits of a data word. See figure 15 for the representative schematic of the parity checking circuit. Although the parity-checking block is implemented in VHDL, a schematic diagram is easier to understand for this simple circuit. The parity checking circuit is very similar to the parity generation block. In fact the main differences are the addition of the inverter U4 and the AND gate U5. The Parity_Check signal is zero unless the Data_Rdy signal is high and the flip flop output Q is low, which is inverted by U4 to a high. This means that the parity is even, and the end of the word has been reached.

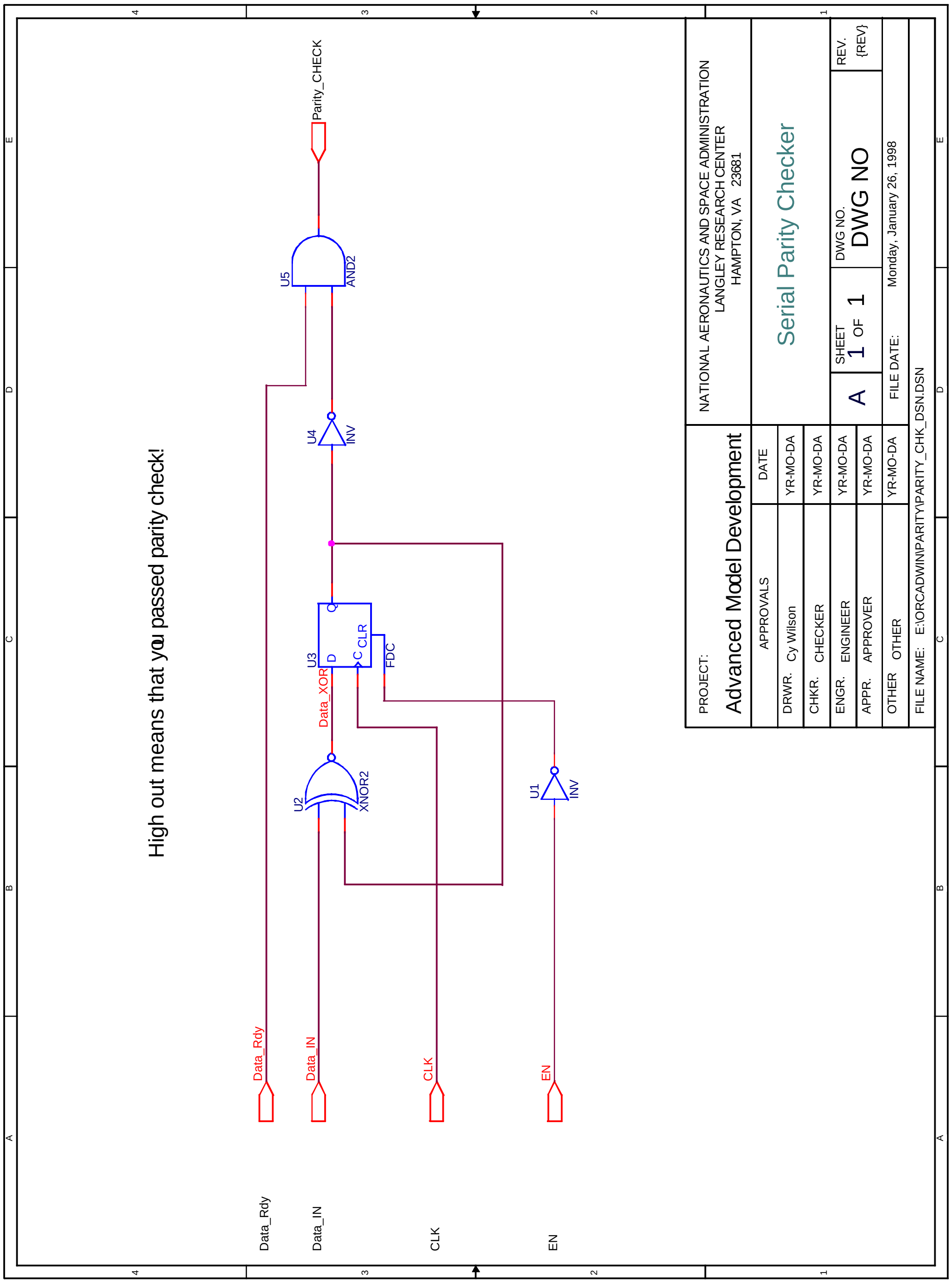


Figure 15 Parity Checking Schematic Diagram

6.3.4.3 Internal Serial Receiver Shift Register

Internally the Shift_En signal is used as an enable for the shift register. While the Shift_En signal is high, the shift register will shift data into the register using the rising edge of the 1 MHz clock. The data bits are latched using rising edge 1 MHz clock. This will ensure that latching will take place in as close to the middle of the bit times as is possible the data is passed to the shift register on the falling edge of the 1 MHz clock. The Shift_En also enables a counter that counts the bits as the are shifted in. When the count reaches 19 the Data_Rdy signal goes high and the count is reset to zero. The SIO_IN_Reg will use the Data_Rdy signal to load the 20 bits of data in parallel into a register. For detailed timing refer to figure 16 for the timing diagram of the SIO receiver.

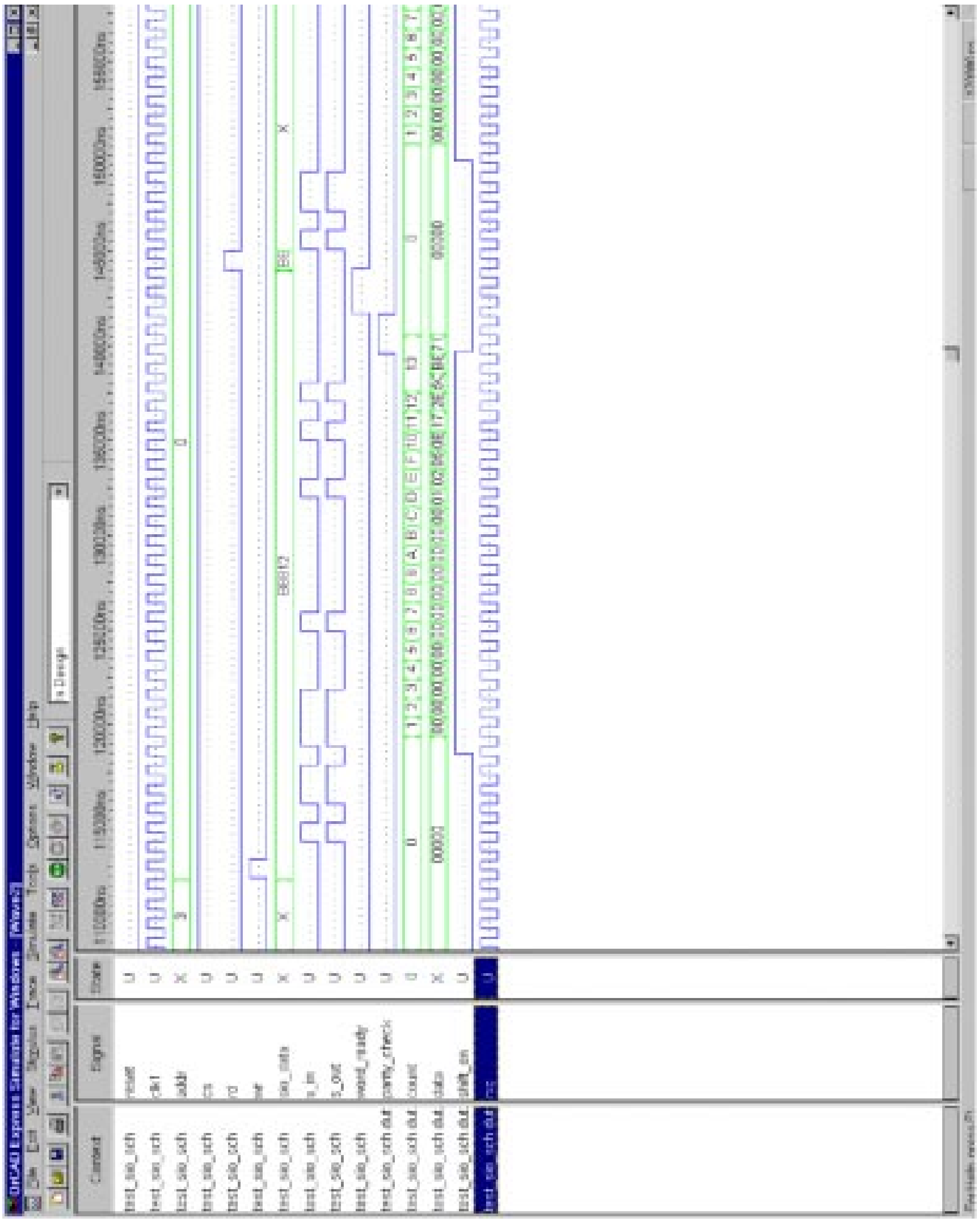


Figure 16   SIO Receiver Timing Diagram

6.3.4.4 Internal Serial Receiver Register

This register provides a temporary storage location for received data. This double buffering scheme allows data to be held in the Internal Serial Receiver register (SIO_IN_REG) while the shift register is shifting in the next data word. This register is 20 bits wide and is latched by the Data_Rdy signal that comes from the shift register.

6.4 External Serial Input/Output

The external serial I/O block is very similar to the internal SIO block. The blocks are so similar that it is more concise to discuss the differences. The main differences are the addition of the FIFO interface to the external SIO block, and the larger number of bits in the bit stream of the external SIO block. See figure 17 for a block diagram of the XSIO design. Also, different clocks are used by the two separate blocks.

The internal SIO block uses a word size of 24 bits including the start bits and parity bit. The external SIO block uses a word size of 28 bits. The extra four bits are the cluster ID number. Since the external, SIO block has a larger word size the input and output registers are 24 bits instead of the 20 bits found in the internal SIO block. The shift registers are also four bits larger for the external SIO block. Naturally, the Add_Start and the Find_Start blocks have their counter values extended by four bits to accommodate the 28-bit word size. The parity generation and parity check blocks are identical in both the internal and external SIO blocks.

The external SIO block has a transit FIFO buffer that is not found in the internal SIO block. The reason for the FIFO is to create a place for data to be stored from the multiple internal SIO blocks. Since the SIO blocks are receive data asynchronously it is possible for all of the blocks to receive data at the same time. When this happens the Word_Ready signals from each of the SIO blocks would go high to indicate that data has been received. Next, the main state machine would write the data received into the FIFO. The main state machine will first get the data from SIO#1, SIO#2, SIO#3 and SIO#4 in that order. The Decode state machine within the XSIO monitors the FIFO_Empty signal. This will drop low if the FIFO is written into. While the FIFO_Empty the decode state machine will read data from the FIFO and transfer it to the XSIO_OUT_Reg so it can be transmitted to the host computer. Since the XSIO is sixteen times faster than the SIO blocks, it should be able to transmit all of the data received before the SIO block can fill the FIFO. The receiver side of the external SIO block does not have a FIFO, or a FIFO interface. Since the receiver is used only for commands and small amounts of configuration data it was deemed unnecessary for it have its own FIFO. The SIO blocks cannot receive commands or data any faster than 1Mbps; therefore, the host should be restrained to meet with this limitation.

The XSIO has one other difference from the SIO blocks and that is the frequency of operation. The SIO block uses the 8 MHz and 1 MHz clocks, while the XSIO uses the 64Mhz and the 16MHz clocks. In both cases, the faster clock is used for the state machine, and the slower clock is used for clocking data in and out.

Other than the three differences mentioned here, the internal and external SIO blocks are essentially identical.

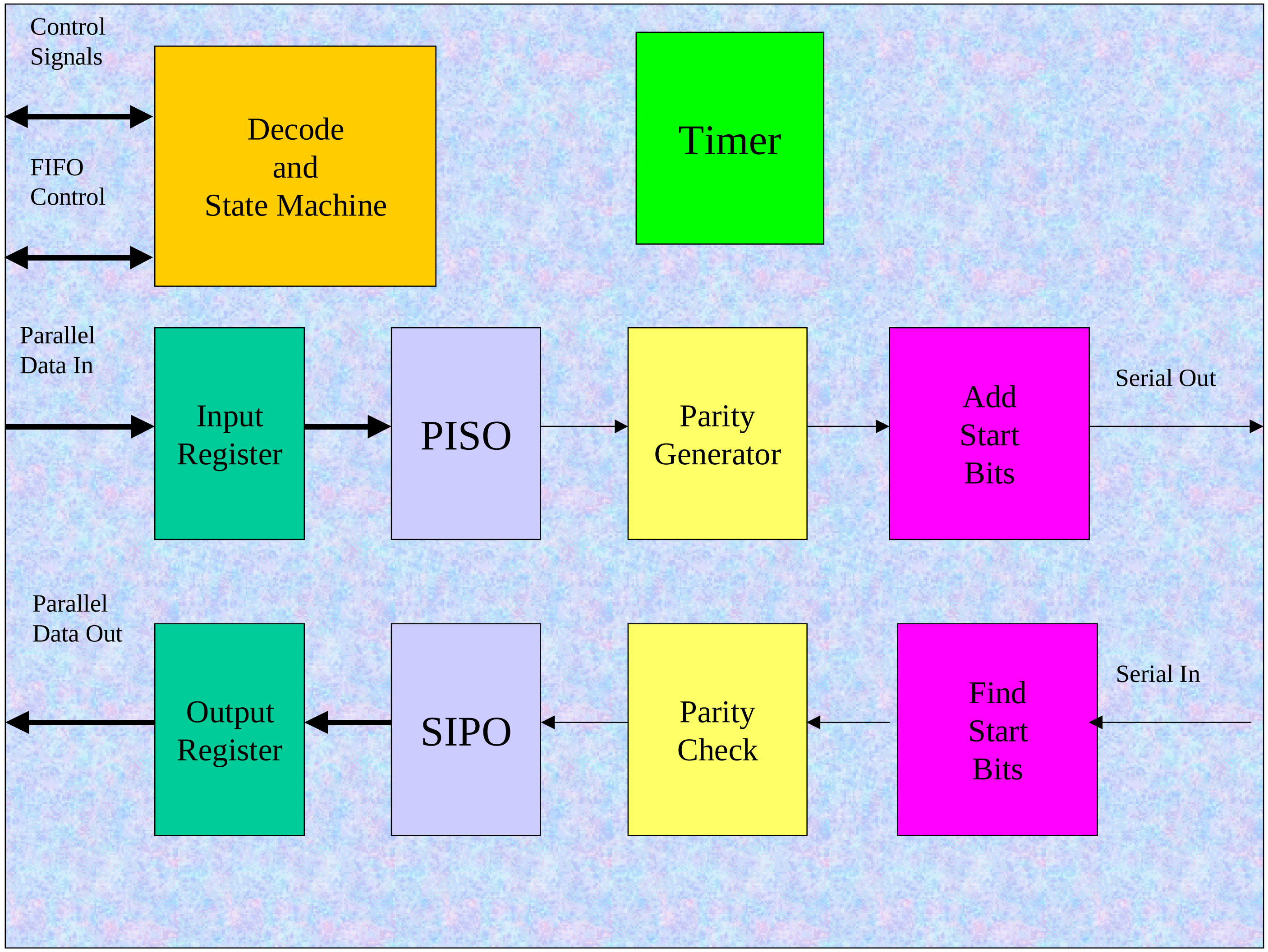


Figure 17 External Serial I/O Block Diagram

6.5 DATA FIFO

The data FIFO is a 16 x 24 bit First In First Out buffer. The purpose of the FIFO is to allow the internal SIO block to have a place to store data received if all of the channels are receiving data at the maximum rate simultaneously. The FIFO is actually made up of two sub-blocks. The control block holds the combinational logic for the FIFO, while the RAM24 block acts as the storage area. The RAM24 is one of the few blocks in the design that uses instantiated components. The RAM24 block is a 16 x 24-bit register file made up of 24 separate 16 x 1-bit ram components. The Xilinx component name is RAM16X1S. The RAM16X1S uses the look up table available within the Xilinx CLB (Combinational Logic Block) to create a 16 x 1 bit register file. The RAM24 block will need to be replaced with a more standard memory component in an ASIC design.

6.6 Clock Generator Block

The clock generator block is the block where the system oscillator comes in to be divided down and distributed. The system oscillator frequency is 64 MHz. It is divided down to provide 16 MHz, 8 MHz, 1 MHz, and 250 kHz. The main state machine and the external SIO blocks use the 64 MHz and the 16 MHz. The internal SIO blocks use the 8 MHz and the 1 MHz. The 250 kHz is used only in the delay block. The 16 MHz, 8 MHz, 1 MHz, and 250 kHz clock lines are used internally to the FPGA and therefore have to be placed on low skew clock buffer lines. In the prototype, this accomplished by placing the signals on Secondary Global Buffers that are found in the Xilinx FPGA chip.

6.7 Bus Interface Block

The bus interface block is the only destination for data that is received from the host telemetry system with a cluster ID of 14. The data portion of this word will be either an address or data for the bus interface's data or address register. If either the address_lo or the data_reg_lo are selected then 16 bits of data are stored at the location. If the high portion of either register is selected then in the case of the data_reg_hi, 8 bits are stored while the address register will store only 2 bits. See figure 18 for a block diagram of the Bus Interface block, see table 8, for the bus interface registers.

Sensor #'s	**Register**
000	Address_Reg_LO
001	Address_Reg_HI
010	Data_Reg_LO
011	Data_Reg_HI
100	Reserved
101	Reserved
110	Reserved
111	Reserved

Table 8 Bus Interface Register

Note: Data can also be sent to the Host computer using the Data Word and that the communication controller cluster ID.

Examples of the read and write sequences:

Write

Write address.

Write data.

Set WR signal

Read

Write address

Set RD signal.

Latch data at destination.

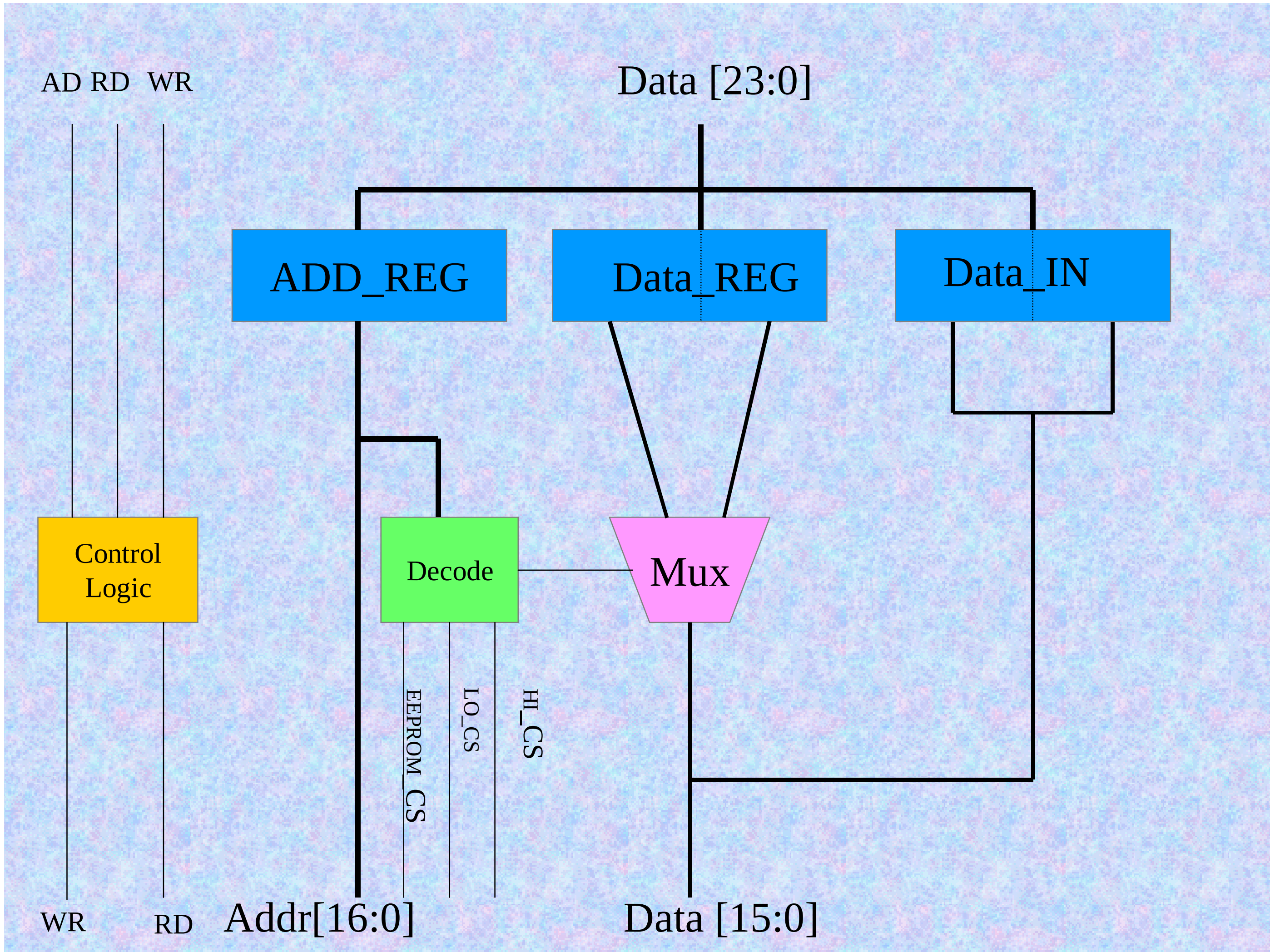


Figure 18 Bus Interface Block Diagram

Chapter 7 Communication Controller ASIC Design

As stated earlier space limitation is the main reason for using ASICs in this project. By using an ASIC, the area for the circuitry can be reduced from the 24.8 square inches of the prototype board. More models will be able to use the system if its size is reduced. While size is the major concern it should be noted that by integrating functions on a single ASIC, power is also reduced. Reducing power reduces the amount of heat dissipated within the model. Since some wind tunnels operate at elevated temperatures, reduced power means reduced cooling needed within the model.

The ASIC process will begin with the VHDL design for the prototype Communication Controller FPGA. Instead of synthesizing this VHDL design for a Xilinx FPGA the code was synthesized using Exemplar's Leonardo tool targeting MOSIS SCMOS standard cell libraries for HP's 0.5um CMOS process.

Chapter 8 Implementation

The communication controller board was developed by way of PC boards. First manufacturer evaluation boards were purchased and tested to insure that the chips chosen had the functionality that was needed. Next, PC boards were routed implementing the communication controller board design. A portion of the communication Controller board design will be used as the design for the communication Controller ASIC. Only after the communication controller board has undergone extensive testing will the ASIC design proceed. Signal integrity and thermal analysis will be accomplished by using software packages such as the Tanner Tools. Fabrication is to occur through the MOSIS brokerage services.

The project has two phases. The first phase was the implementation of the entire system with each module on a separate PC board of size 3U or 3.94” x 6.3”. The board area is 24.8 square inches. The system at this stage consists of a Balance card, a Shear Stress card, two DAS cards, a communication Controller card and a telemetry card. The boards that are being used in this phase are called prototype boards. If scheduling permits, this phase of the design will be tested in a wind tunnel model.

In the second phase, the prototype boards are to be reduced in size using ASIC technology. Two of the cards will have DAS and sensor ASICs residing together on a card. The communication Controller ASIC will reside on a separate card in close proximity to the telemetry card.

The prototype PC boards incorporate all of the discrete chips that will be incorporated into one ASIC chip. The boards are six layers with the two internal layers used for power and ground planes. This leaves the other four layers for signals. Complete functional testing will include testing at operational speed.

After testing the prototype boards, the VHDL will be re-synthesized for the ASIC version. The synthesis portion of design process will use the SCMOS standard cell library instead of the Xilinx FPGA library. “Synthesis is a powerful tool for translating a design from the HDL level to the gate level. Before HDLs became prevalent, requirement specifications were manually translated directly to the gate level. Automating gate-level implementations by using synthesis saves designers time and reduces human error.”[6] After synthesis comes place and route, which will yield the layout for the ASIC. “Only after you complete the layout do you know the parasitic capacitance and therefore the delay associated with the interconnect. This postroute delay information must be returned to the schematic in a process known as back-annotation.”[7] Simulations performed at this level should verify the functionality and timing of the design. The design process is an iterative one where any problems found that cause the layout to be modified will also cause new simulations to be run, and re-tested for problems. Error checking will be performed before the finished layout is transferred to MOSIS for fabrication.

While the ASICs are being fabricated, boards for the ASICs will have to be routed and then fabricated. These are the boards used in phase three of the project.

Chapter 9 Testing

Before the prototype board testing the Communication Controller FPGA was simulated using a testbench. "The test bench applies stimuli to the module under test, compares the output from the module to expected responses, and reports and discrepancies found."[8] After the simulations had tested the functionality of the FPGA; the FPGA was synthesized, placed, and routed. Portions of the design were simulated using the backannotated timing information from the place and route software.

The prototype board was tested using standard test equipment, such as a logic analyzer, multimeters, oscilloscopes, and voltage sources. The board will be tested for operational conditions as well as for maximum speed of operation. Testing included input values that are outside of the operational range to insure that the system did not latch up or cause problems if these values are encountered at any time.

"Use of RAM-based rapidly reprogrammable FPGAs allows special FPGA configurations to be developed for use in system tests. These can be used to test board traces and even adjacent non-FPGA devices as well as the FPGA chips themselves. This technique has the potential to reduce test time for completed assemblies and the need for complex board testers and to allow self-test in the field."[9]

The EEPROM was tested using the test header placed on the communication Controller board. This header allows connections to the

EEPROM's address, data lines, and control signals. The header has control signals for the FPGA to allow power to be supplied to the board with the FPGA held in a reset state. This allows the testing of the EEPROM without affecting the rest of the circuitry on the communication Controller board. Once the wiring was determined to be working correctly, the EEPROM was loaded with the configuration for the FPGA chip. After loading, the data was read back one last time; to insure that it had been loaded properly. At this stage, the FPGA was allowed to configure itself from the information placed into the EEPROM chip. From this point forward, testing proceeded to check the functions of the communication Controller board.

The phase I testing used the communication controller board in a stand-alone fashion, with only buttons or switches as stimulus. Since this board finished fabrication first, it had to be tested by itself. Next, the wiring of the board to each of the communication interfaces was verified. This test included the driver chips for both the internal and external communications. The clock divider block was tested next; this test was also passed. The SIO portion of the design was tested next, using the clock divider block. After several iterations, all of the bugs were removed from the SIO design, and it passed the functional test. The XSIO block and clock divider blocks were then tested together. The XSIO required some changes, but it too was able to pass the functional tests. At this stage, 51 of the 55 blocks (counting each copy separately) have passed functional testing, or roughly 92% of the VHDL code for the system. This concludes phase I testing. See figure 19 for the phase I testing configuration.

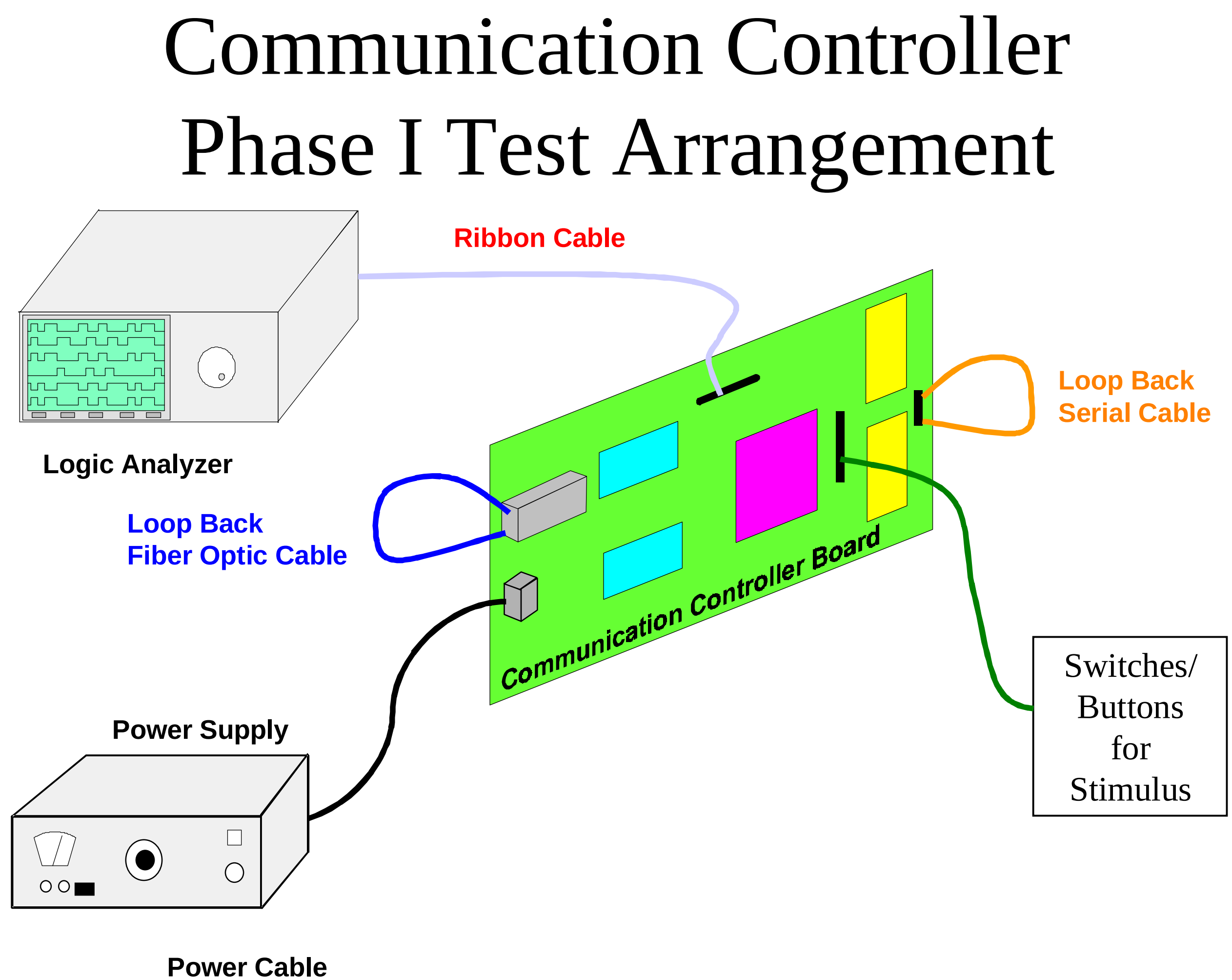


Figure 19 Communication Controller Board Phase I Test Diagram

To perform phase II testing of the communication Controller board the following will be necessary: Two computers, one to act as the host and one to act as a DAS simulator. A power supply that is capable of delivering up to one amp at +8V is needed. Fiber optic cables for external communication, and internal communication wire cables. The internal simulator board configured for simulating a DAS module, and the host interface board. A logic analyzer, and a voltmeter for system debugging. See figure 20, below, for a diagram of the communication Controller board test arrangement.

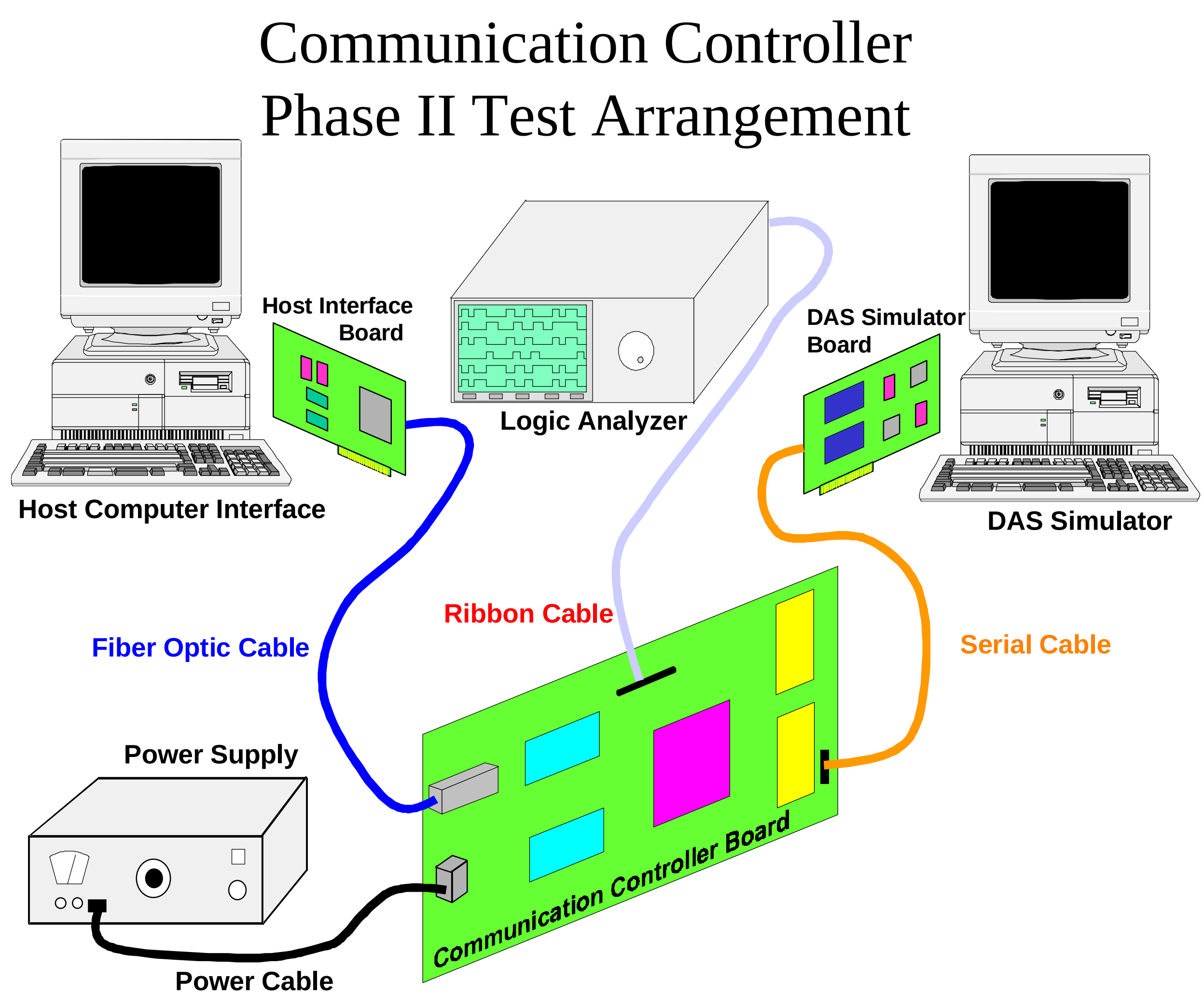


Figure 20 Communication Controller Board Phase II Test Diagram

After phase I testing, it will be necessary to attach a serial interface card. The serial interface card will be a generic card with a PC interface (ISA bus) and four serial interfaces that match those found in the internal communication scheme. One of these boards will be fabricated. The serial interface card will be used in testing the communication Controller board, to act as a DAS module, while at other times it will be used in testing of the DAS module and will act as a communication Controller board. Again, the use of these boards will allow testing to occur in parallel. The use of the interface cards will allow the PC to handle all communications across the serial lines. Then only software would differentiate whether a DAS module or the communication Controller module is being simulated.

One PC will be used as test equipment at this stage of testing. It will act as both the telemetry card and as the host computer. In this manner it will display the information sent across the fiber optic cable by the communication Controller card and will send data across the fiber optic cable in a manner that will mimic the host computer. See figure 21 for a diagram of the integration test arrangement.

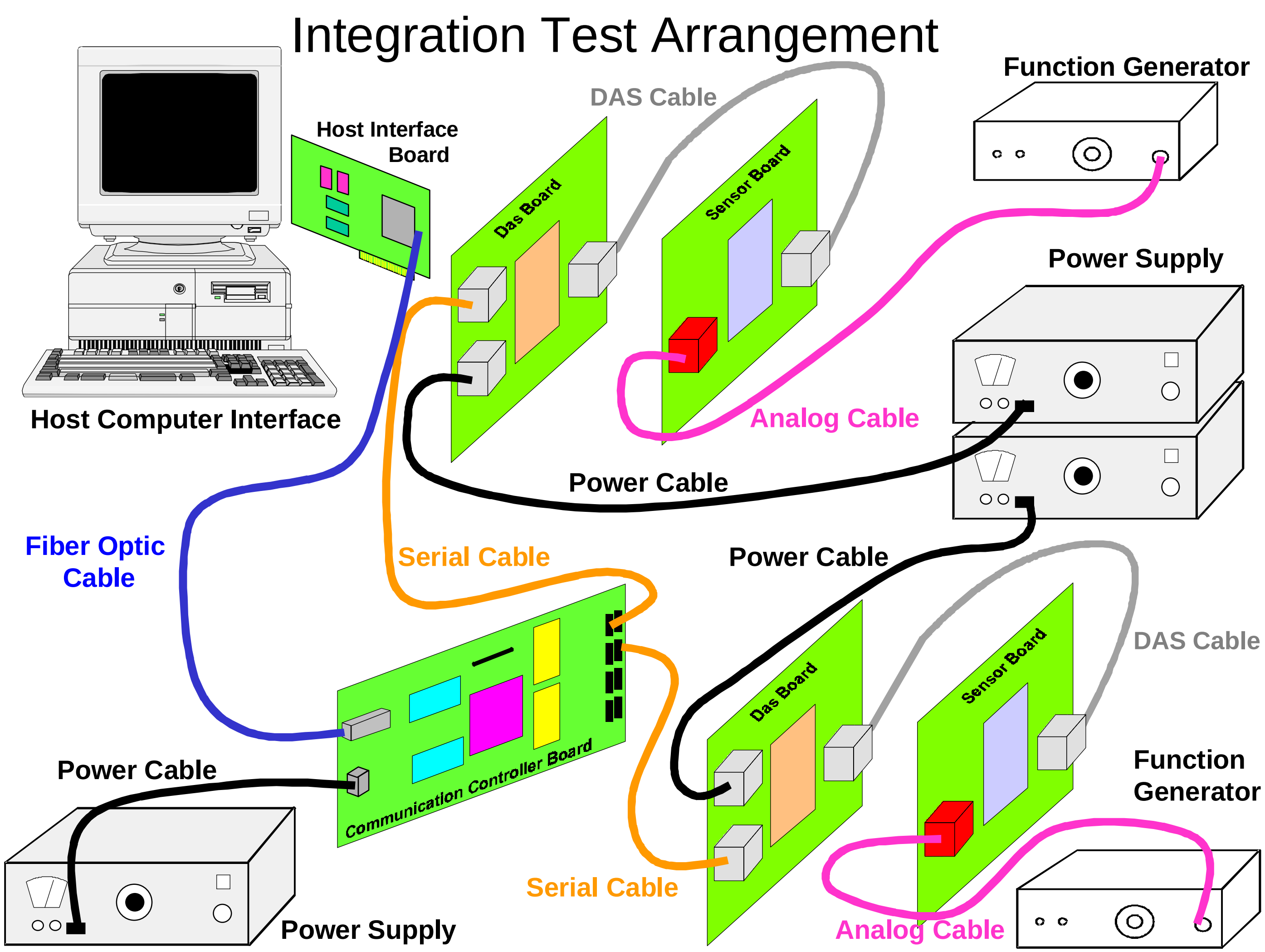


Figure 21 Integration Test Diagram

The ASIC will be tested using vectors from the logic analyzer. A test fixture will be fabricated to allow easy connection of digital input signals, as well as the output signals. Communication ports will be checked using differential drivers/receivers located on the test fixture board. These will convert the signal back to digital levels for either the logic analyzer or external computer test equipment to transfer data and commands.

The test using the logic analyzer will be performed using the generic PGA (Pin Grid Array) test board. This board is a simple design that centers around an 84 pin ZIF (Zero Insertion Force) socket. On the board, each pin of the socket is taken out to a header/connector for the logic analyzer. On the way to the connector, each signal trace will have a test point post. The test points can be used for signal measurement or injection by external test equipment. In between the test, points and the connector are jumper posts. The jumpers will allow signals to be isolated from the logic analyzer if necessary. Separate posts will be provided for the power connections of the logic analyzer probes to the DUT (Device Under Test).

The board is very versatile due to its generic nature. The board will be used in testing of all of the ASICs developed by the Model Instrumentation Project. Each ASIC will be placed in the ZIF socket and the jumpers will be removed for each of the power pins associated with the ASIC.
Next power connections will be made to the test point posts for which a jumper was removed. Any other test equipment can be added at this stage, scopes and voltage meters can be attached to test points, while current measurements will

necessitate the removal of jumpers and the use of the jumper posts for their connections. Last, the logic analyzer connectors will be seated and testing may begin.

Testing of the final board will include tests that cover the full operational temperature range of the board. These tests will check functionality and performance at both ends of the temperature envelope. Functional testing after vibrational loads that exceed the maximum found in operation will also be performed.

All of the VHDL was simulated using the ORCAD simulator. This was to verify the functionality with unit delays. The phase I testing is complete. The phase II testing has begun. Within the blocks that have undergone testing, the most common problem found was the clocking of data on the same edge of the clock on which data is changing. Simple functional simulations will not discover this type of problem. Backannotation simulations will find this problem because real routing delays are used. Only portions of the design would yield a backannotation simulation, due to a problem involving the use of the global reset signal. Because an overall backannotation simulation was not available, the system will require more thorough testing and debugging.

Another problem area uncovered during testing is the delay that signals exhibit due to routing. Xilinx chips have large routing delays because of the use active elements for programmable routing. In one particular signal, the routing delays accounted for 57% of the signals delay, while the propagation delay only accounted for 43%. This caused problems in the Main State machine where the clock rates were high, and there was large amounts of decode logic. The state transitions would glitch through unwanted states as the signals would propagate through the decode logic with different amounts of routing delays. There are several ways to remedy this type of problem. One way is to use one hot state encoding. This would reduce the number of state bits that are changing and would have the effect of increasing the maximum speed at which the state machine could operate. The method chosen, however, was to reduce the size of the state machine by using logic that is more combinational. This method gave greater reductions in the size of the design while it reduced the amount of glitching and increased the maximum speed at which the design could operate.

Chapter 10 Conclusions

Due to the size of the internal serial I/O blocks only four channels instead of 12 could fit within the FPGA. This problem could be overcome in some small measure by further optimization of the SIO block. This might allow the number of internal channels to grow to 6, but not to 12. This course of action may be followed in the name of efficiency, but is not necessary for the ASIC version. ASICs can be fabricated in any size, so increasing the channels to 12 would increase the area of the die and its cost. The area the design takes up can be optimized further in the Xilinx chip by reducing the states within the state machines. This can be done by using logic that is more combinational in place of state machine decoding.

Appendix A Hardware Diagrams and Part Lists

Prototype Communication Controller Board Top

Prototype Communication Controller Board Bottom

Prototype Communication Controller Board Schematic Diagram

Communication Controller FPGA Schematic Diagram

Internal Communication Block Schematic Diagram

External Communication Block Schematic Diagram

Communication Controller Board Parts List

Internal Simulator Board Schematic Diagram

Host Communication Board Schematic Diagram

Communication Controller Xilinx XC4025E FPGA Pin Locations

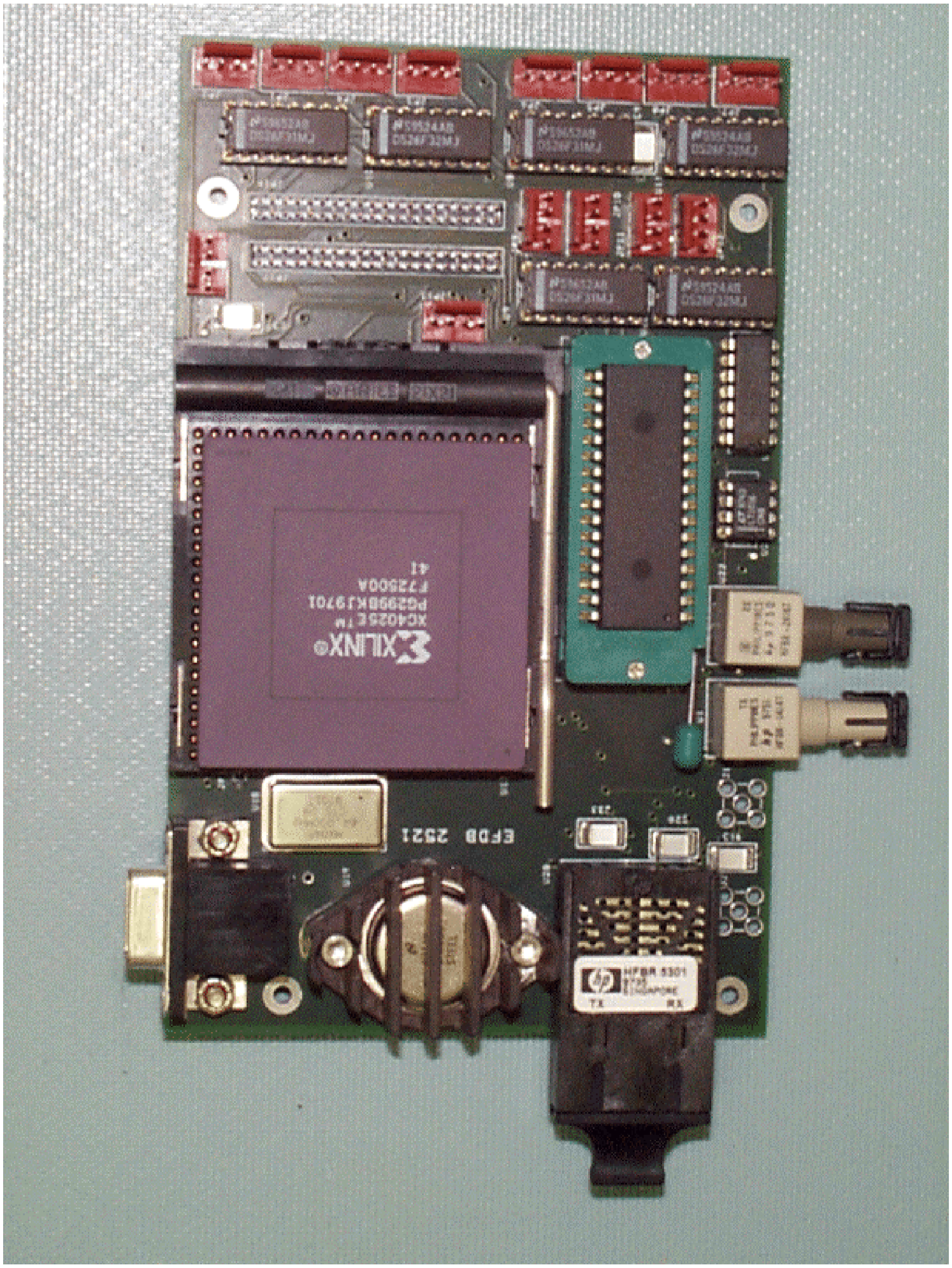


Figure 22   Communication Controller Top

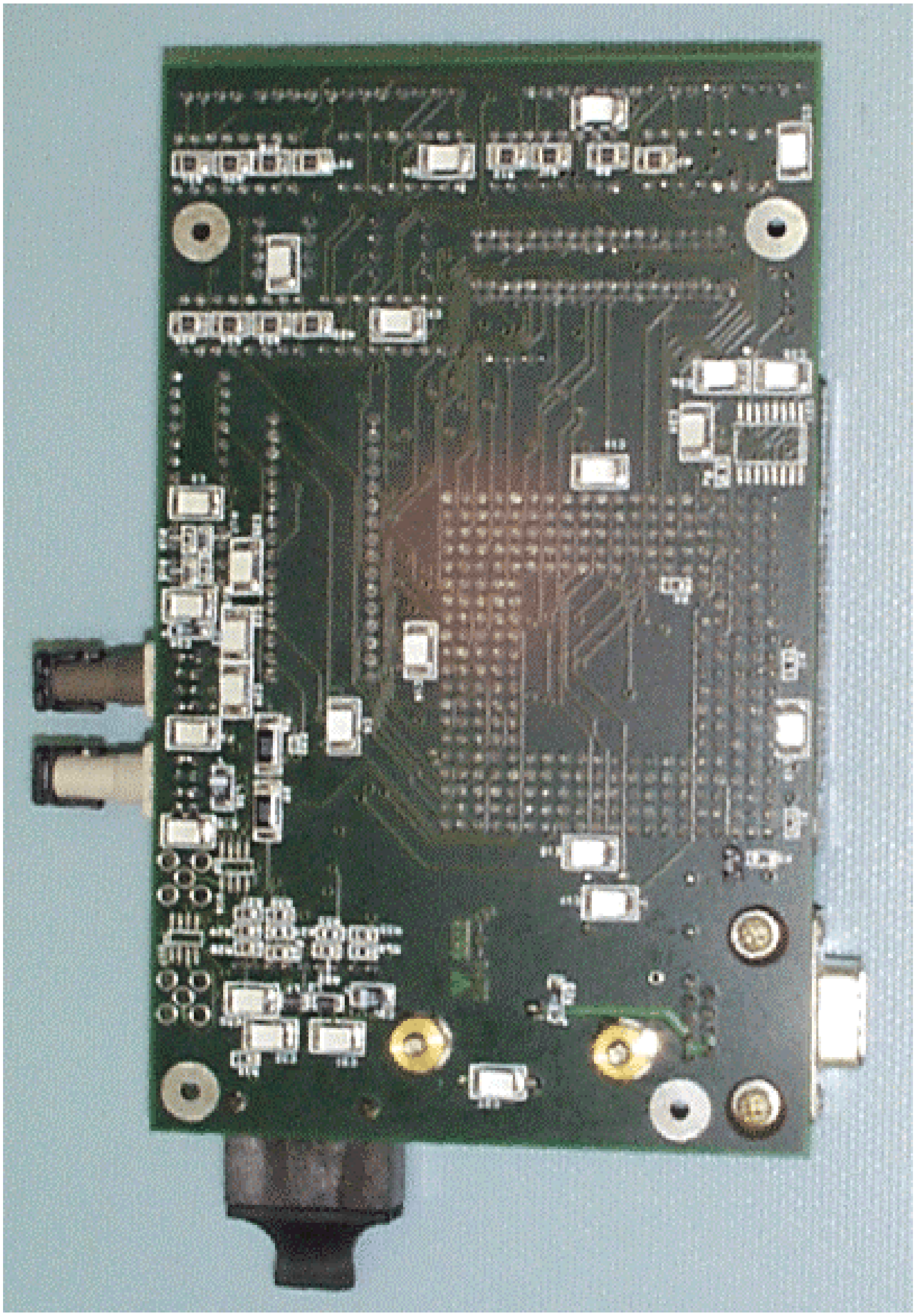


Figure 23   Communication Controller Bottom

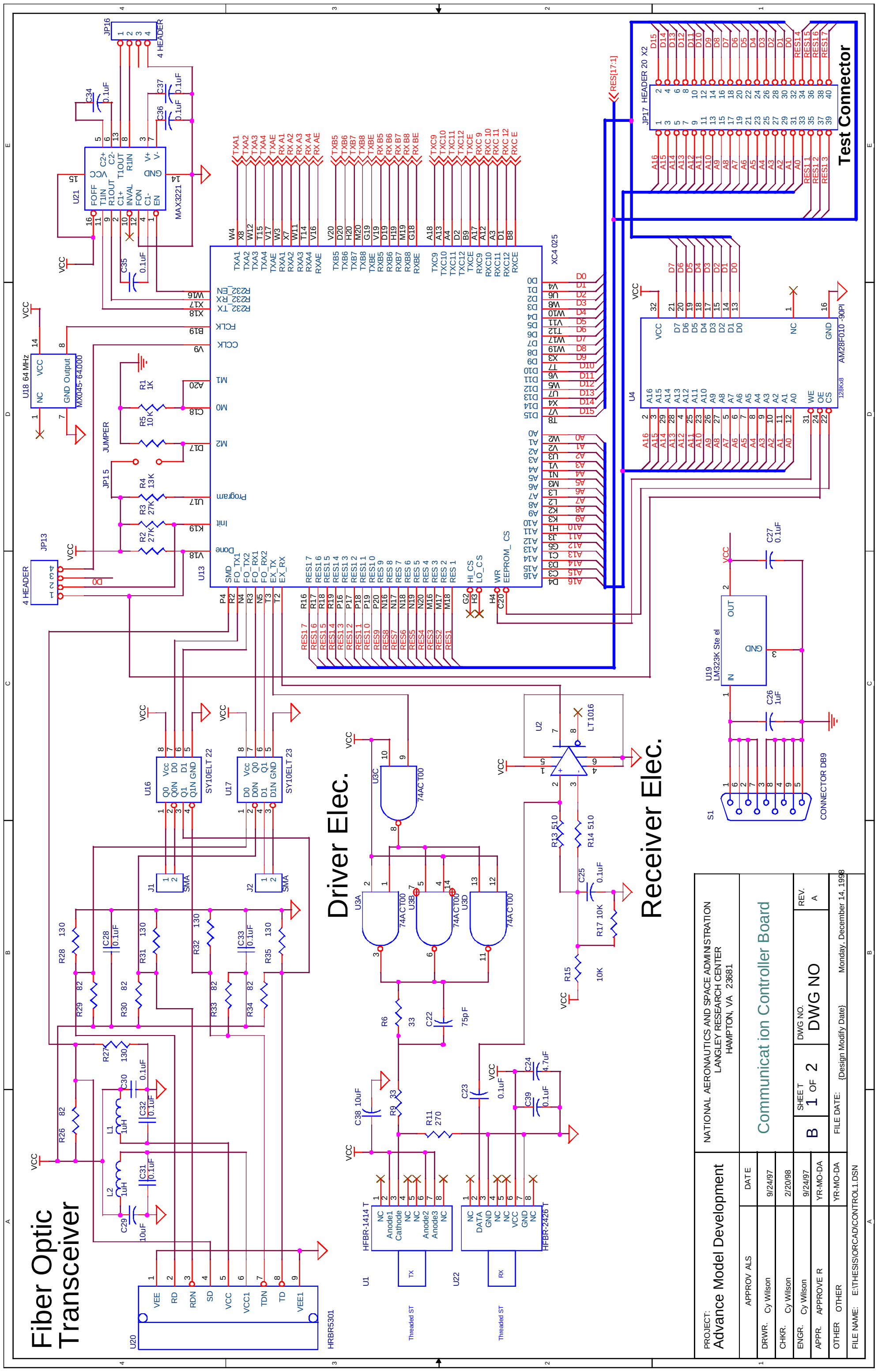


Figure 24-A Communication Controller Schematic Diagram

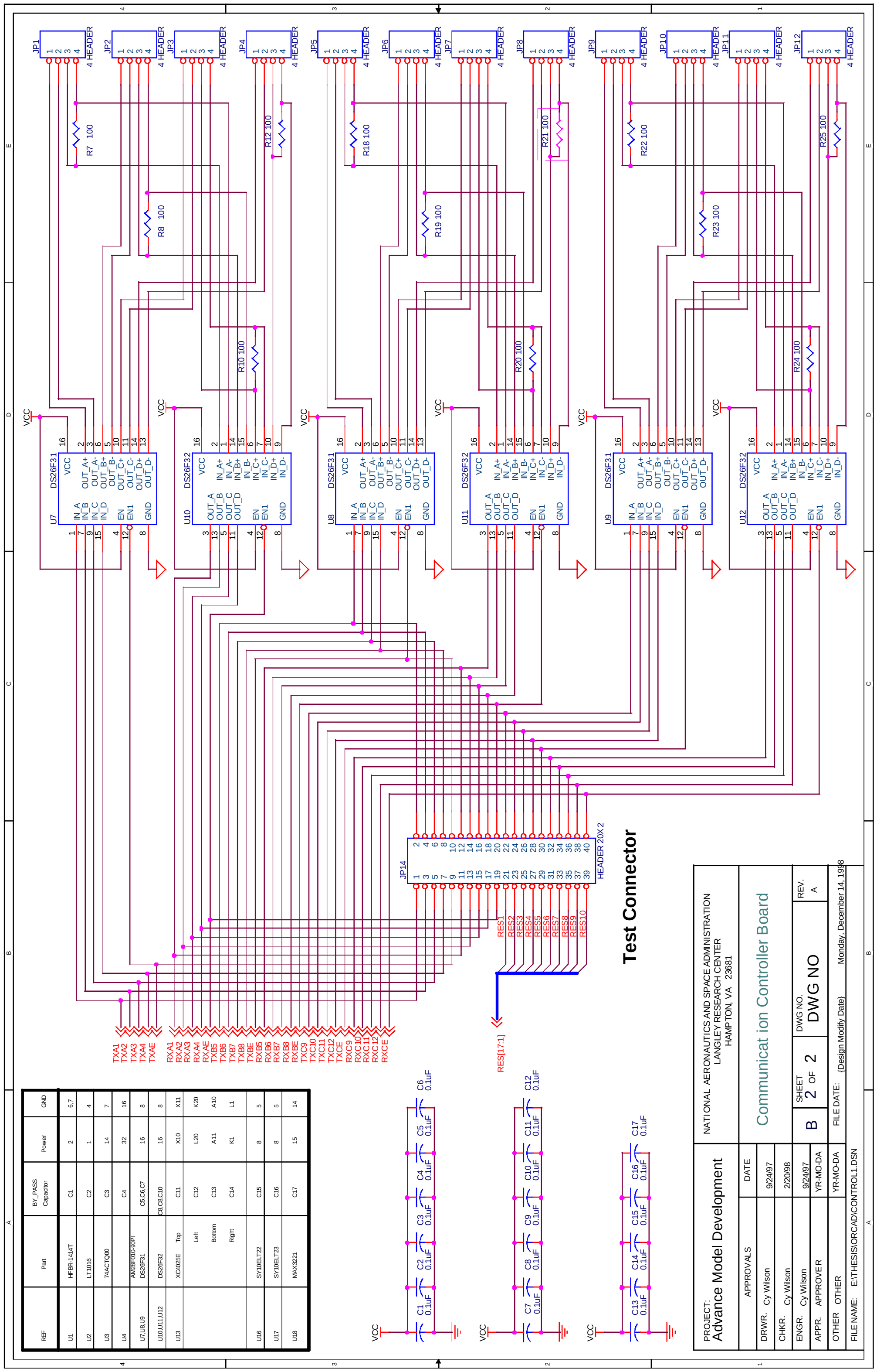


Figure 24-B Communication Controller Schematic Diagram

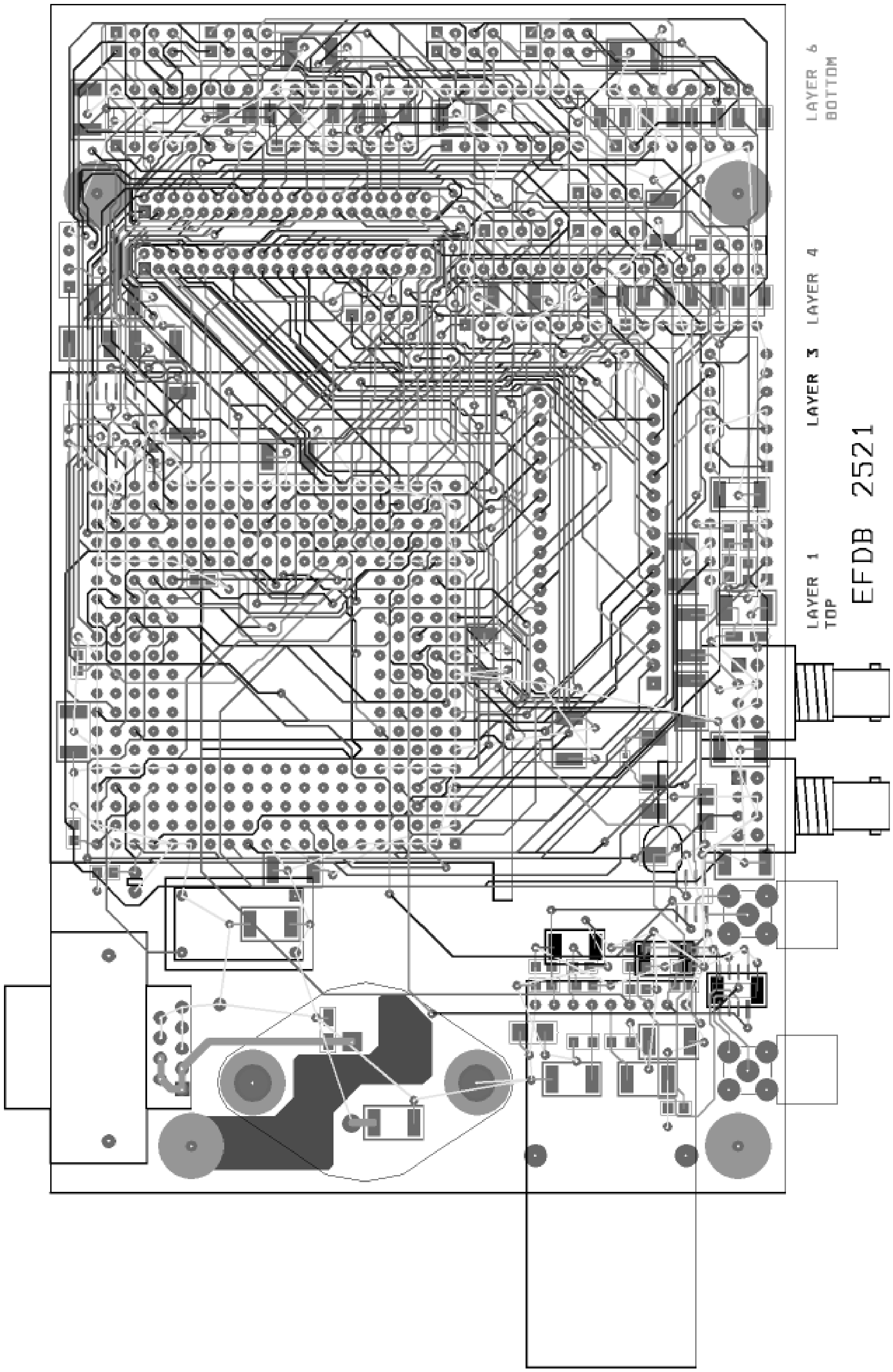


Figure 25 Communication Controller Board Layout

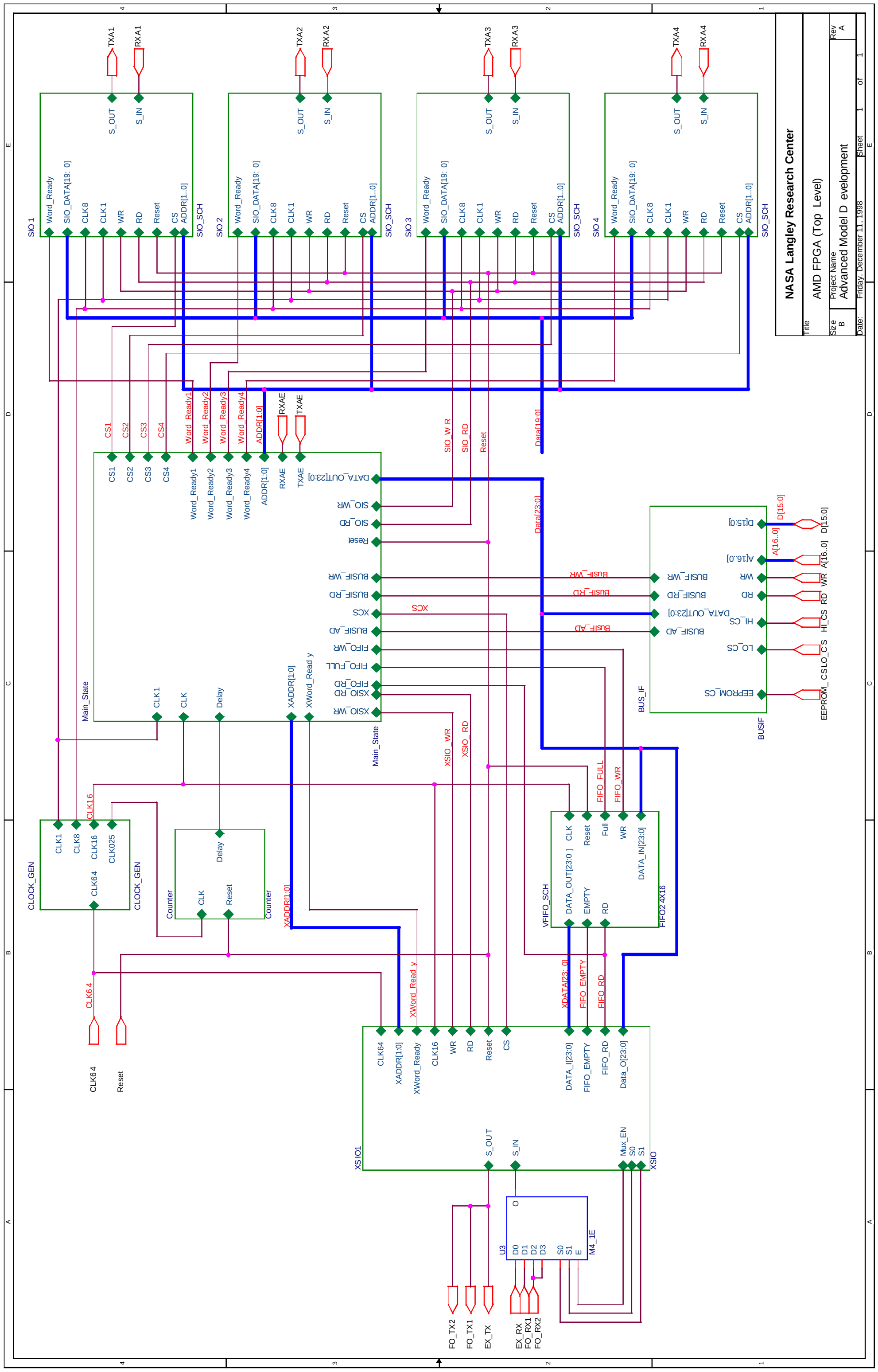


Figure 26 Communication Controller FPGA Schematic Diagram

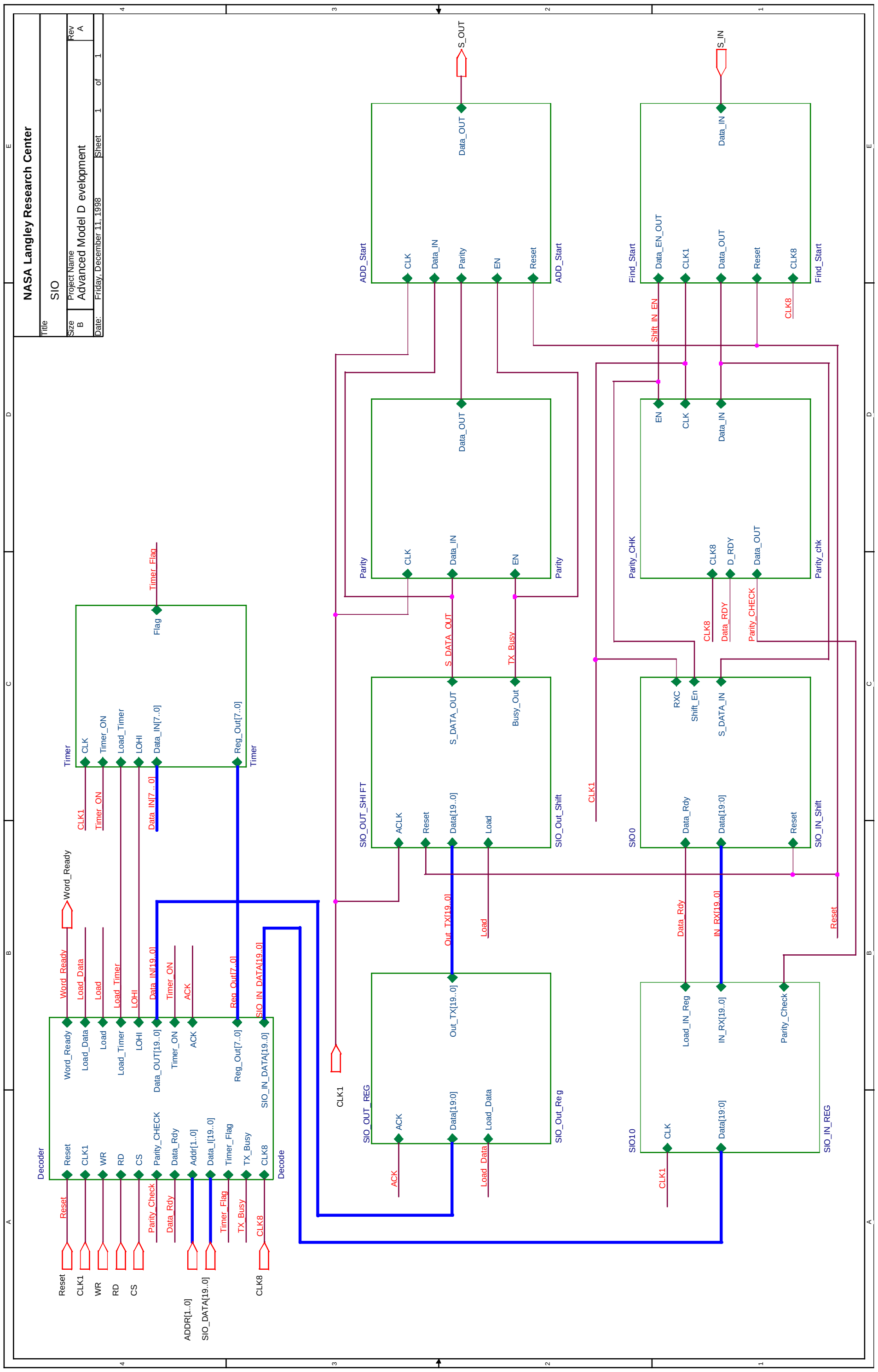


Figure 27   Internal Communication Block Diagram

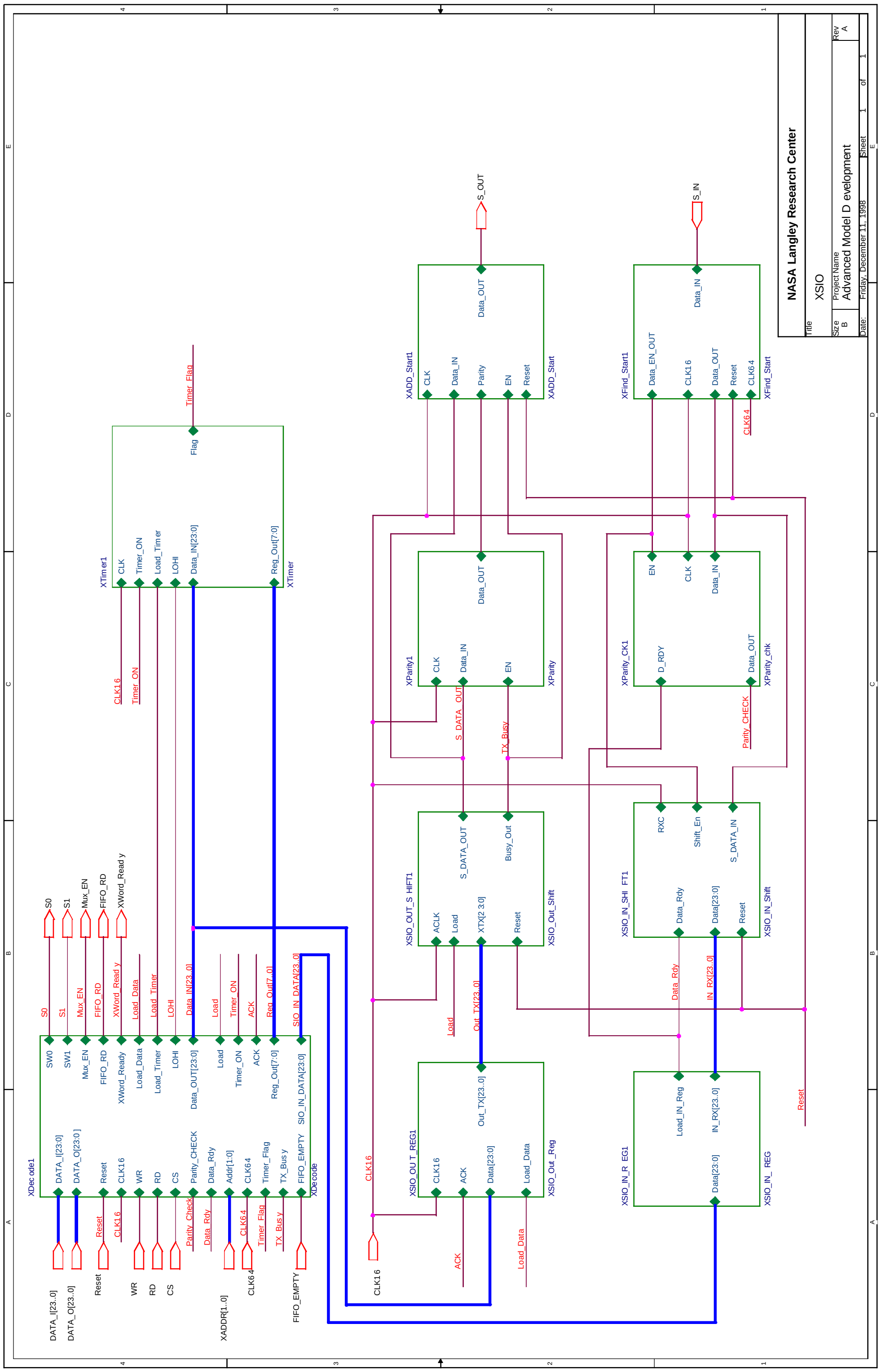


Figure 28   External Communication Block Diagram

Communication Controller Board Parts List

DESIGN	E:\ORCADWIN\AMD3\CONTROL1.DSN			
PAGE1		LOGICAL		
HEADER	**ID**	**Part Reference**	**Value**	**PCB Footprint**
PART	INS14179	R1	1K	SM/R_0805
PART	INS134936	U4	AM28F010-90PI	DIP.100/32/W.600/ZIF
PART	INS707543	JP4	4 HEADER	SIP/TM/L.400/4
PART	INS707547	JP2	4 HEADER	SIP/TM/L.400/4
PART	INS707549	JP1	4 HEADER	SIP/TM/L.400/4
PART	INS708460	U12	DS26F32	DIP.100/16/W.300/L.775
PART	INS718589	R9	33	SM/R_2512
PART	INS718591	C38	10uF	SM/CT_3528_12
PART	INS718595	C22	75pF	SM/C_0402
PART	INS718861	R6	33	SM/R_2512
PART	INS719541	R12	100	SM/R_1210
PART	INS719932	R13	510	SM/R_0805
PART	INS720505	C27	0.1uF	SM/C_1913
PART	INS720507	C26	1uF	SM/CT_3216_12
PART	INS720509	U19	LM323K Steel	TO3N
PART	INS720511	S1	CONNECTOR DB9	CONNDBRT\9S
PART	INS722579	R7	100	SM/R_1210
PART	INS707545	JP3	4 HEADER	SIP/TM/L.400/4
PART	INS728153	R8	100	SM/R_1210
PART	INS706402	U10	DS26F32	DIP.100/16/W.300/L.775
PART	INS729080	R10	100	SM/R_1210
PART	INS731151	JP8	4 HEADER	SIP/TM/L.400/4
PART	INS731153	JP6	4 HEADER	SIP/TM/L.400/4
PART	INS731155	JP5	4 HEADER	SIP/TM/L.400/4
PART	INS731157	R21	100	SM/R_1210
PART	INS731159	R18	100	SM/R_1210
PART	INS731161	JP7	4 HEADER	SIP/TM/L.400/4
PART	INS731163	R19	100	SM/R_1210
PART	INS731165	R20	100	SM/R_1210
PART	INS731474	JP12	4 HEADER	SIP/TM/L.400/4
PART	INS731476	JP10	4 HEADER	SIP/TM/L.400/4
PART	INS731478	JP9	4 HEADER	SIP/TM/L.400/4
PART	INS731480	R25	100	SM/R_1210
PART	INS731482	R22	100	SM/R_1210
PART	INS731484	JP11	4 HEADER	SIP/TM/L.400/4
PART	INS731486	R23	100	SM/R_1210
PART	INS731488	R24	100	SM/R_1210
PART	INS734845	R17	10K	SM/R_0805
PART	INS734847	C25	0.1uF	SM/C_1913
PART	INS734849	R15	10K	SM/R_0805
PART	INS731825	R14	510	SM/R_0805
PART	INS738399	C24	4.7uF	SM/CT_3216_12
PART	INS739780	U3D	74ACT00	DIP.100/14/W.300/L.775

HEADER	ID	Part Reference	Value	PCB Footprint
PART	INS739876	U3A	74ACT00	DIP.100/14/W.300/L.775
PART	INS744121	R11	270	SM/R_0805
PART	INS739828	U3B	74ACT00	DIP.100/14/W.300/L.775
PART	INS720045	C23	0.1uF	SM/C_1913
PART	INS777473	U16	SY10ELT22	SOG.050/8/WG.244/L.225
PART	INS788745	JP15	JUMPER	JUMPOST2
PART	INS801614	J1	SMA	RF/SMA/R
PART	INS5188	C31	0.1uF	SM/C_1913
PART	INS5248	C30	0.1uF	SM/C_1913
PART	INS5787	L2	1uH	SM/L_1008
PART	INS5894	C32	0.1uF	SM/C_1913
PART	INS5896	L1	1uH	SM/L_1008
PART	INS6485	R26	82	SM/R_0805
PART	INS6733	C29	10uF	SM/CT_3528_12
PART	INS802932	R28	130	SM/R_0805
PART	INS7186	R31	130	SM/R_0805
PART	INS7188	C28	0.1uF	SM/C_1913
PART	INS7808	C33	0.1uF	SM/C_1913
PART	INS7806	R35	130	SM/R_0805
PART	INS7804	R32	130	SM/R_0805
PART	INS803637	U20	HRBR5301	HFBR-5XXX
PART	INS68461	C13	0.1uF	SM/C_1913
PART	INS68342	C3	0.1uF	SM/C_1913
PART	INS68439	C5	0.1uF	SM/C_1913
PART	INS7182	R30	82	SM/R_0805
PART	INS68340	C2	0.1uF	SM/C_1913
PART	INS7800	R33	82	SM/R_0805
PART	INS68457	C16	0.1uF	SM/C_1913
PART	INS243457	C7	0.1uF	SM/C_1913
PART	INS68453	C11	0.1uF	SM/C_1913
PART	INS7180	R29	82	SM/R_0805
PART	INS68443	C8	0.1uF	SM/C_1913
PART	INS241841	C9	0.1uF	SM/C_1913
PART	INS68445	C6	0.1uF	SM/C_1913
PART	INS68459	C14	0.1uF	SM/C_1913
PART	INS241843	C10	0.1uF	SM/C_1913
PART	INS68344	C4	0.1uF	SM/C_1913
PART	INS68455	C15	0.1uF	SM/C_1913
PART	INS67934	C1	0.1uF	SM/C_1913
PART	INS68451	C12	0.1uF	SM/C_1913
PART	INS7802	R34	82	SM/R_0805
PART	INS775225	C17	0.1uF	SM/C_1913
PART	INS18243	R2	27K	SM/R_0805
PART	INS731823	U2	LT1016	DIP.100/8/W.300/L.475
PART	INS109837	U18	MX045-64.000	OSC\4P
PART	INS739023	U3C	74ACT00	DIP.100/14/W.300/L.775
PART	INS14177	R5	10K	SM/R_0805
PART	INS18239	R4	13K	SM/R_0805
PART	INS18241	R3	27K	SM/R_0805
PART	INS850273	JP13	4 HEADER	SIP/TM/L.400/4

HEADER	ID	Part Reference	Value	PCB Footprint
PART	INS871957	U1	HFBR-1414T	HFBR-X41XT
PART	INS708262	U8	DS26F31	DIP.100/16/W.300/L.775
PART	INS708462	U9	DS26F31	DIP.100/16/W.300/L.775
PART	INS706406	U7	DS26F31	DIP.100/16/W.300/L.775
PART	INS889420	JP16	4 HEADER	SIP/TM/L.400/4
PART	INS892614	C34	0.1uF	SM/C_1913
PART	INS892775	C36	0.1uF	SM/C_1913
PART	INS892777	C37	0.1uF	SM/C_1913
PART	INS894853	C35	0.1uF	SM/C_1913
PART	INS872116	U22	HFBR-2426T	HFBR-X41XT
PART	INS889422	U21	MAX3221	SOG.050/16/WG.420/L.400
PART	INS883467	JP17	HEADER 20X2	BCON2MM/2X20
PART	INS777475	U17	SY10ELT23	SOG.050/8/WG.244/L.225
PART	INS906312	C39	0.1uF	SM/C_1913
PART	INS909443	R27	130	SM/R_0805
PART	INS798761	JP14	HEADER 20X2	BCON2MM/2X20
PART	INS708260	U11	DS26F32	DIP.100/16/W.300/L.775
PART	INS801612	J2	SMA	RF/SMA/R
PART	INS859017	U13	XC4025	PGA\299P\ZIF

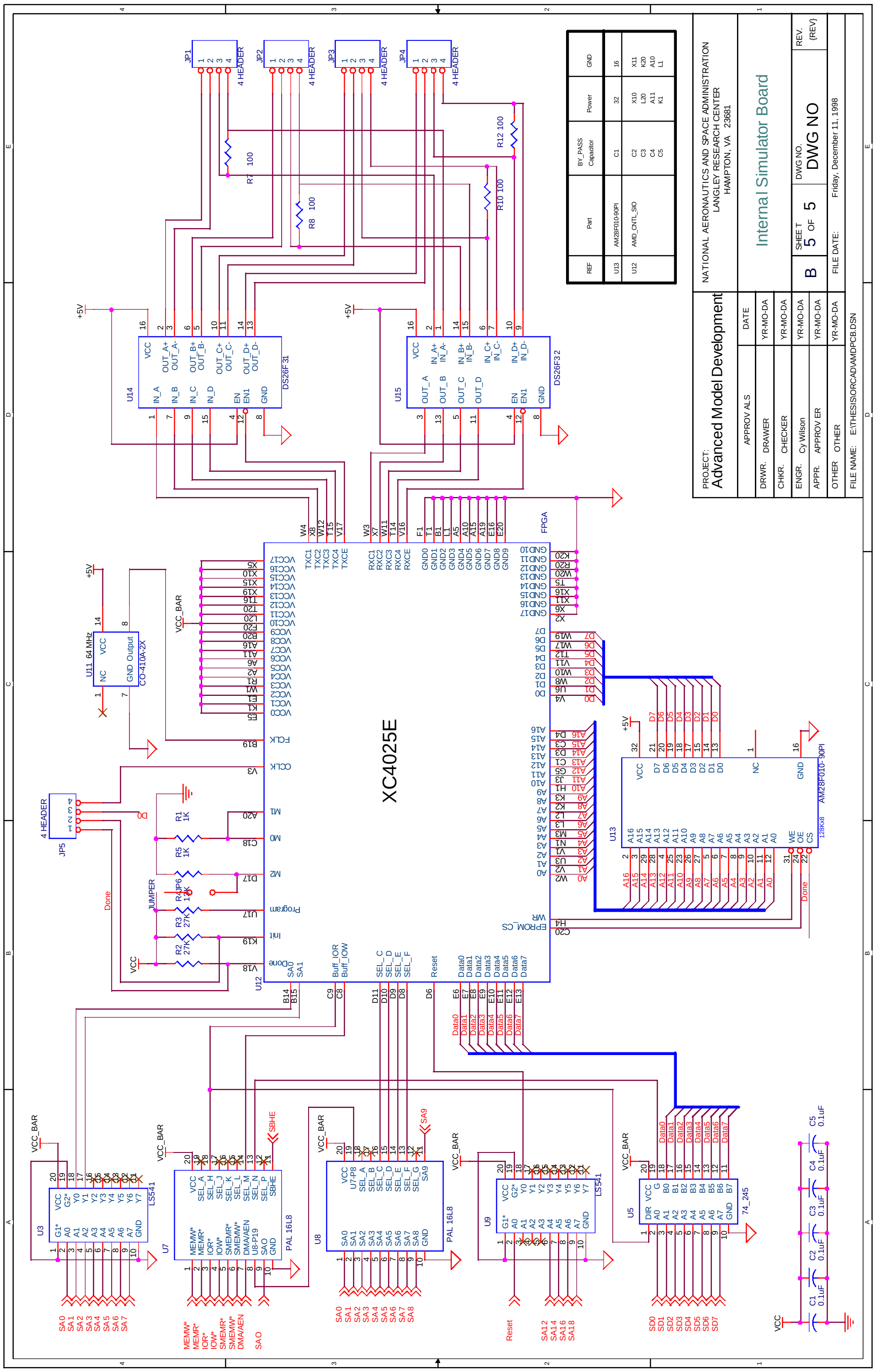


Figure 29 Internal Simulator Board Schematic Diagram

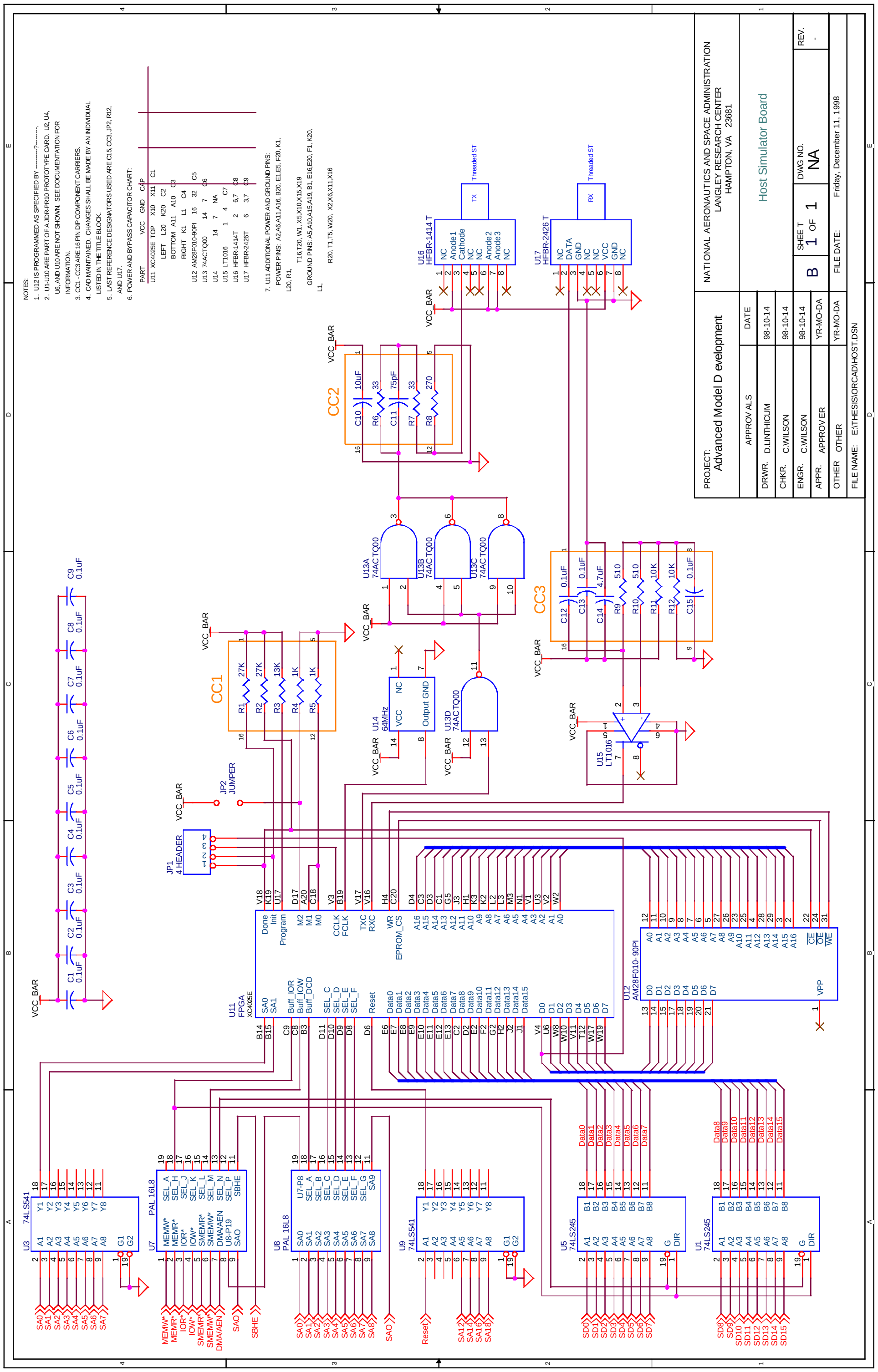


Figure 30 Host Board Schematic Diagram

Communication Controller Xilinx XC4025E FPGA Pin Locations

XC4025E Pad	**PG 299**	**Board Name**	**XC4025E Pad**	**PG 299**	**Board Name**
VCC	K1	-	VCC	E5	-
I/O (A8)	K2	A8	I/O (A17)	B2	
I/O (A9)	K3	A9	I/O	B3	
I/O (A19)	K5		I/O	E6	
I/O (A18)	K4		I/O TDI	D5	TDI
I/O	J1		I/O TCK	C4	TCK
I/O	J2		I/O	A3	RXC11
I/O (A10)	H1	A10	I/O	D6	
I/O (A11)	J3	A11	I/O	E7	
I/O	J4		I/O	B4	
I/O	J5		I/O	C5	
I/O	H2	CS0	I/O	A4	TXC11
I/O	G1		I/O	D7	
VCC	E1	-	I/O	C6	
I/O	H3	LO_CS	I/O	E8	
I/O	G2	HI_CS	I/O	B5	
I/O	H4	WR	GND	A5	-
I/O	F2		I/O	B6	
GND	F1	-	I/O	D8	
I/O	H5		I/O TMS	C7	TMS
I/O	G3		I/O	B7	
I/O	D1	RXC12	VCC	A6	-
I/O	G4		I/O	C8	
I/O	E2		I/O	E9	
I/O	F3		I/O	A7	
I/O (A12)	G5	A12	I/O	D9	
I/O (A13)	C1	A13	I/O	B8	RXCE
I/O	F4		I/O	A8	
I/O	E3		I/O	C9	
I/O	D2	TXC12	I/O	B9	TXCE
I/O	C2		I/O	D10	
I/O	F5		I/O	C10	
I/O	E4		GND	A10	-
I/O (A14)	D3	A14	VCC	A11	-
I/O SGCK1 GCK8 (A15)	C3	A15/CLK025	I/O	B10	
VCC	A2	-	I/O	B11	
GND	B1	-	I/O	C11	
I/O PGCK1 GCK1 (A16)	D4	A16	I/O	E11	

XC4025E Pad	PG 299	Board Name	XC4025E Pad	PG 299	Board Name
I/O	D11		I/O	G17	
I/O	A12	RXC10	I/O	F18	
I/O	B12		I/O	H16	
I/O	A13	TXC10	I/O	E19	
I/O	C12		I/O	F19	
I/O	D12		GND	E20	-
I/O	E12		I/O	H17	
I/O	B13		I/O	G18	RXBE
VCC	A16	-	I/O	G19	TXBE
I/O	A14		I/O	H18	
I/O	C13		VCC	F20	-
I/O	B14		I/O	J16	
I/O FCLK2	D13		I/O	G20	
GND	A15	-	I/O	J17	
I/O	B15		I/O	H19	RXB7
I/O	E13		I/O	H20	TXB7
I/O	C14		I/O	J18	
I/O	A17	RXC9	I/O	J19	
I/O	D14		I/O	K16	
I/O	B16		I/O	J20	
I/O	C15		I/O	K17	
I/O	E14		I/O	K18	
I/O	A18	TXC9	I/O (INIT)	K19	INIT
I/O	D15		VCC	L20	-
I/O	C16		GND	K20	-
I/O	B17		I/O	L19	
I/O	B18		I/O	L18	
I/O	E15		I/O	L16	
I/O	D16		I/O	L17	
O (M1)	A20	M1	I/O	M20	TXB8
I/O SGCK2 GCK2	C17	CLK16	I/O	M19	RXB8
GND	A19	-	I/O	N20	
I (M0)	C18	M0	I/O	M18	
VCC	B20	-	I/O	M17	
GND	E16	-	I/O	M16	
I (M2)	D17	M2	I/O	N19	
I/O PGCK2 GCK3	B19	FCLK	I/O	P20	
I/O (HDC)	C19		VCC	T20	-
I/O	F16		I/O	N18	
I/O	E17		I/O	P19	
I/O	D18		I/O	N17	
I/O (LDC)	C20	EEPROM	I/O	R19	
I/O	F17		GND	R20	-
I/O	G16		I/O	N16	
I/O	D19	RXB6	I/O	P18	

XC4025E Pad	PG 299	Board Name	XC4025E Pad	PG 299	Board Name
I/O	E18		I/O	U20	
I/O	D20	TXB6	I/O	P17	
I/O	T19		I/O (CS0)	X14	
I/O	R18		I/O	U12	
I/O	P16		I/O	W13	
I/O	V20	TXB5	I/O	X13	
I/O	R17		I/O	V12	
I/O	T18		I/O	W12	TXA3
I/O	U19		I/O	T11	
I/O	V19	RXB5	I/O	X12	
I/O	R16		I/O	U11	
I/O	T17		I/O (D4)	V11	D4
I/O	U18		I/O	W11	RXA3
I/O SGCK3 GCK4	X20	CLK8	VCC	X10	-
GND	W20	-	GND	X11	-
DONE	V18	DONE	I/O (D3)	W10	D3
VCC	T16	-	I/O (RS)	V10	
VCC	X19	-	I/O	T10	
PROGRAM	U17	PROGRAM	I/O	U10	
I/O (D7)	W19	D7	I/O	X9	
I/O PGCK3 GCK5	W18		I/O	W9	
I/O	T15	TXA4	I/O	X8	TXA2
I/O	U16		I/O	V9	
I/O	V17	TXAE	I/O	U9	
I/O	X18	R232_TX	I/O	T9	
I/O	U15		I/O (D2)	W8	D2
I/O	T14	RXA4	I/O	X7	RXA2
I/O(D6)	W17	D6	VCC	X5	-
I/O	V16	RXAE	I/O	V8	
I/O	X17	R232_RX	I/O FCLK4	W7	
I/O	U14		I/O	U8	
I/O	V15		I/O	W6	
I/O	T13		GND	X6	-
I/O	W16	R232_EN	I/O	T8	D15
I/O	W15		I/O	V7	D14
GND	X16	-	I/O	X4	D13
I/O	U13		I/O	U7	D12
I/O	V14		I/O	W5	D11
I/O,FCLK3	W14		I/O	V6	D10
I/O	V13		I/O	T7	D9
VCC	X15	-	I/O	X3	D8
I/O (D5)	T12	D5	I/O (D1)	U6	D1

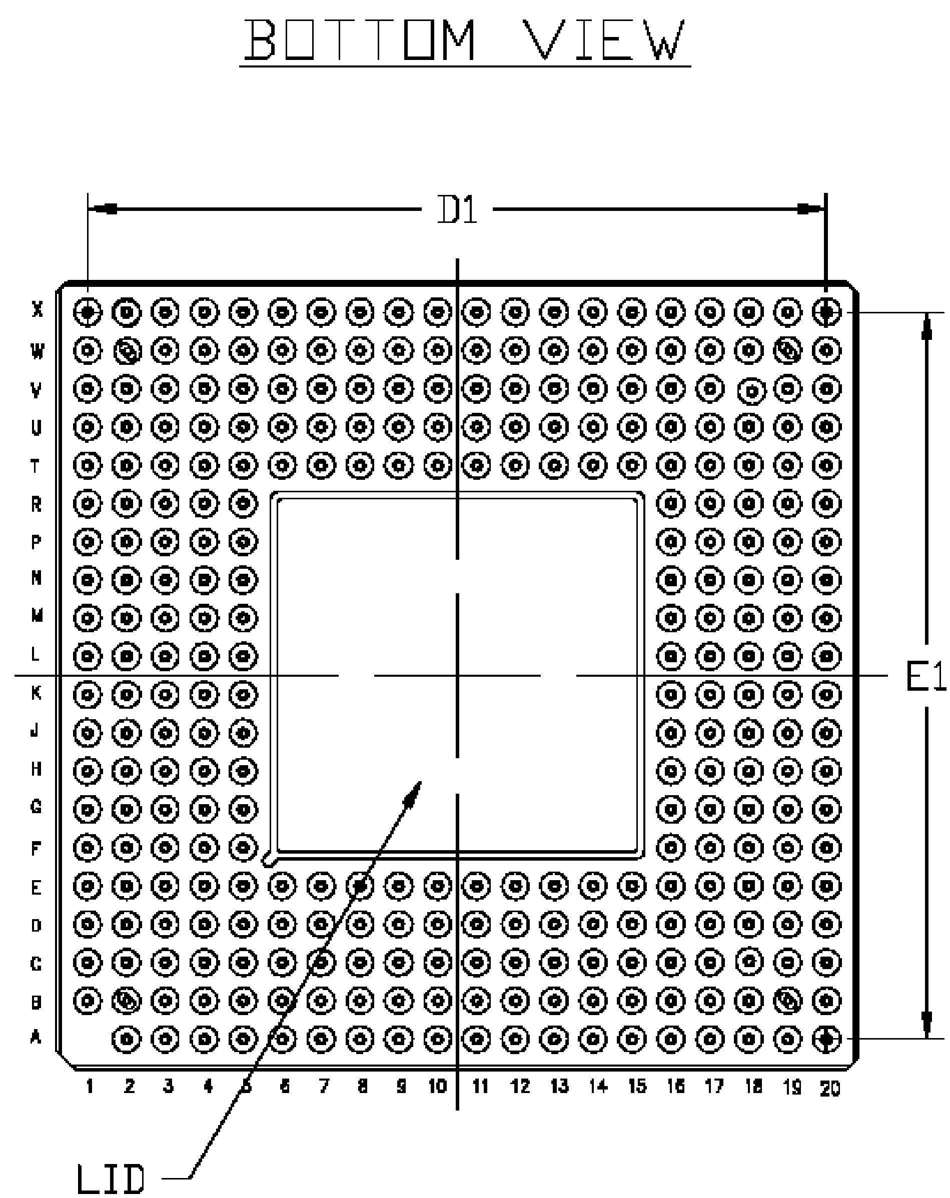
XC4025E Pad	**PG 299**	**Board Name**	**XC4025E Pad**	**PG 299**	**Board Name**
I/O (RCLK RDY/BUSY)	V5		I/O	R3	FO_RX1
I/O	W4	TXA1	I/O	N5	FO_RX2
I/O	W3	RXA1	I/O	T2	EX_RX
I/O	T6		I/O	R2	FO_TX1
I/O	U5		GND	T1	-
I/O(D0,DIN)	V4	D0	I/O	N4	FO_TX2
I/O SGCK4 GCK6 (DOUT)	X1	DOUT/CLK1	I/O	P3	
CCLK	V3	CCLK	I/O	P2	
VCC	W1	-	I/O	N3	
GND	T5	-	VCC	R1	-
O TDO	U4	TD0	I/O	M5	
GND	X2	-	I/O	P1	
I/O (A0 WS)	W2	A0	I/O	M4	
I/O PGCK4 GCK7 (A1)	V2	A1	I/O	N2	
I/O	R5		I/O(A4)	N1	A4
I/O	T4		I/O(A5)	M3	A5
I/O (CS1 A2)	U3	A2	I/O	M2	
I/O (A3)	V1	A3	I/O	L5	
I/O	R4		I/O(A21)	M1	A21
I/O	P5		I/O(A20)	L4	A20
I/O	U2		I/O(A6)	L3	A6
I/O	T3	EX_TX	I/O(A7)	L2	A7
I/O	U1		GND	L1	-
I/O	P4	SMD			

Pin Locations

Top		Bottom	
X	All	A	All
W	2-19	B	2-19
V	3-18	C	3-18
U	4-17	D	4-17
T	5-16	E	5-16

Right (From Top View)		Left (From Top View)	
B	1	B	20
C	1,2	C	19,20
D	1,2,3	D	18,19,20
E	1,2,3,4	E	17,18,19,20
F	1,2,3,4,5	F	16,17,18,19,20
G	1,2,3,4,5	G	16,17,18,19,20
H	1,2,3,4,5	H	16,17,18,19,20
J	1,2,3,4,5	J	16,17,18,19,20
K	1,2,3,4,5	K	16,17,18,19,20
L	1,2,3,4,5	L	16,17,18,19,20
M	1,2,3,4,5	M	16,17,18,19,20
N	1,2,3,4,5	N	16,17,18,19,20
P	1,2,3,4,5	P	16,17,18,19,20
R	1,2,3,4,5	R	16,17,18,19,20
T	1,2,3,4	T	17,18,19,20
U	1,2,3	U	18,19,20
V	1,2	V	19,20
W	1	W	20

Data Bus is on Top
Address Bus is (Top, Right, and Bottom) call it Top



Power (From bottom View)

	VCC	GND
Top	T16,X19,X15,X10,X5	T5,X16,X11,X6,X2
Left	E5,K1,E1,W1,R1	F1,T1,B1,L1
Right	B20,F20,L20,T20	E20,K20,R20,W20
Bottom	A2,A6,A11,A16	A5,A10,A15,A19,E16

Figure 31 Xilinx 299 PGA Pin Diagram

References

[1] Ashenden, Peter J, The Designer's Guide to VHDL Morgan Kaufmann Publishers, Inc. San Francisco 1996. p609.

[2] Navabi, Zainalabedin VHDL Analysis and Modeling of Digital Systems McGraw-Hill New York, 1993. p22.

[3] Bhasker, Jayaram A VHDL Primer Prentice Hall Englewood Cliffs, 1992 p34.

[4] Trimberger, Stemphen M., Field Programmable Gate Array Technology Kluwer Academic Publishers Boston/ Dordrecht/London, 1994. p2.

[5] Navabi, Zainalabedin VHDL Analysis and Modeling of Digital Systems McGraw-Hill New York, 1993. p200.

[6] Wall, James, Macdonald, Anne, The NASA ASIC Guide Draft 0.6 Jet Propulsion Laboratory Pasadena 1993 p255.

[7] Smith, Michael John Sebastian, Application Specific Integrated Circuits Addison-Wesley Reading/Harlow/Menlo Park/Berkley/Don Mills/Sydney/Amsterdam/Tokyo/Mexico City 1997 p345.

[8] Schoen, Joel M. Performance and Fault Modeling with VHDL Prentice Hall Englewood Cliffs, 1992, p225.

[9] Oldfield, John V., Dorf, Richard C., Field Programmable Gate Arrays Reconfigurable Logic for Rapid Prototyping and Implementation of Digital Systems John Wiley & Sons, Inc. New York/Chicherster/Brisbane/ Toronto/Singapore, 1995 p85.

Index

REPORT DOCUMENTATION PAGE			*Form Approved* *OMB No. 0704-0188*

1. AGENCY USE ONLY (*Leave blank*)

2. REPORT DATE
March 1999

3. REPORT TYPE AND DATES COVERED
Technical Memorandum

4. TITLE AND SUBTITLE
Design of the Wind Tunnel Model Communication Controller Board

5. FUNDING NUMBERS
WU 519-20-21-01

6. AUTHOR(S)
William C. Wilson

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
NASA Langley Research Center
Hampton, VA 23681-2199

8. PERFORMING ORGANIZATION REPORT NUMBER
L-17816

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
National Aeronautics and Space Administration
Washington, DC 20546-0001

10. SPONSORING/MONITORING AGENCY REPORT NUMBER
NASA/TM-1999-209108

11. SUPPLEMENTARY NOTES
The information presented in this report was offered as a thesis in partial fulfillment of the requirements for the Degree of Master of Science in Applied Physics and Computer Science, Christopher Newport University, Newport News, Virginia, December 1998.

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Unclassified-Unlimited
Subject Category 33 Distribution: Nonstandard
Availability: NASA CASI (301) 621-0390

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

The NASA Langley Research Center's Wind Tunnel Reinvestment project plans to shrink the existing data acquisition electronics to fit inside a wind tunnel model. Space limitations within a model necessitate a distributed system of Application Specific Integrated Circuits (ASICs) rather than a centralized system based on PC boards. This thesis will focus on the design of the prototype of the communication Controller board. A portion of the communication Controller board is to be used as the basis of an ASIC design. The communication Controller board will communicate between the internal model modules and the external data acquisition computer. This board is based around an FPGA, to allow for reconfigurability. In addition to the FPGA, this board contains buffer RAM, configuration memory (EEPROM), drivers for the communications ports, and passive components.

14. SUBJECT TERMS
VHDL, FPGA, Xilinx, Communication, Fiber Optic, Serial

15. NUMBER OF PAGES
119

16. PRICE CODE
A06

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL